НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СТІРЕНКО____

«___»_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Система електронного документообігу підприємства»

Виконав:
студент IV курсу, групи ІО-61
Колпак Максим Віталійович _____

Керівник:
Доцент кафедри ОТ, д.т.н.,
Сергієнко Анатолій Михайлович _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н.,
Сімоненко Валерій Павлович _____

Рецензент:
Доцент кафедри СКС, к.т.н.,
Орлова Марія Миколаївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій СТІРЕНКО_____
«___»_____2020 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Колпака Максима Віталійовича

1. Тема проєкту «Система електронного документообігу підприємства», керівник проєкту Сергієнко Анатолій Михайлович, д. т. н., доц., затверджені наказом по університету від «07» травня 2020р. №1081-с

2. Термін подання студентом проєкту 26 травня 2020 року

3. Вихідні дані до проєкту: технічна документація, теоретичні дані, розроблена система електронного документообігу.

4. Зміст пояснювальної записки: аналіз існуючих рішень на ринку, проєктування і розробка корпоративної системи електронного документообігу.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

Блок-схема алгоритму валідації форми створення документу(1шт.).

Діаграма бази даних(1шт.).

Структурна схема системи(1шт.).

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сімоненко В.П., проф.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Затвердження теми роботи	01.09.2019	
2	Вивчення та аналіз завдання	01.01.2019	
3	Аналіз існуючих передумов для розробки	28.02.2020	
4	Проектування архітектури системи	20.03.2020	
5	Розробка додатку	25.04.2020	
6	Оформлення матеріалів роботи	22.05.2020	
7	Передзахист	26.05.2020	
8	Захист		

Студент Максим КОЛПАК

Керівник Анатолій СЕРГІЄНКО

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломного проєкту.

Анотація

В даному дипломному проєкті проведено аналіз існуючих рішень на рику корпоративних систем електронного документообігу. Встановлено їх існуючі недоліки. Спроєктовано і розроблено систему електронного документообігу підприємства, яка враховує ці недоліки.

Annotation

This Bachelor's work includes the analysis of the existing enterprise documents management systems. It was highlighted their existing disadvantages. New enterprise document management system were designed and developed to fix this disadvantages.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ІАЛЦ. 467200.000 ВП	Відомість дипломного проєкту	1	
3	А4	ІАЛЦ. 467200.001 ТЗ	Технічне завдання	3	
4	А4	ІАЛЦ. 467200.002 ПЗ	Пояснювальна записка	63	
5	А4	ІАЛЦ. 467200.003 Д1	Блок-схема алгоритму валідації форми створення документу	1	
6	А4	ІАЛЦ. 467200.004 Д2	Діаграма бази даних	1	
7	А4	ІАЛЦ. 467200.005 Д3	Структурна схема системи	1	

Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ. 467200.000 ВП		
		Колпак М.В.			Система електронного документообігу підприємства Відомість дипломного проєкту	Літ.	Арк 1 / Аркушів 1
		Сергієнко А.М.					
Реценз.		Орлова М.М				НТУУ КПІ, ФІОТ, ІО-61	
Н. Контр.		Сімоненко В. П.					

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломного проєкту
освітньо-кваліфікаційного рівня__бакалавр___

на тему: “ Система електронного документообігу підприємства”

Київ – 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується розробку системи електронного документообігу.

Область застосування: корпоративне програмне забезпечення.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить підвищення використання систем електронного документообігу бізнесом.

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкту є розробка системи електронного документообігу підприємства.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література проектування програмного забезпечення, публікації в Інтернеті щодо даної предметної області.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розроблюваного продукту

- Розробка серверної частини додатку
- Розробка клієнтської частини додатку
- Розробка інтерфейсу користувача

5.2. Вимоги до програмного забезпечення

- Операційна система Windows
- .NET Core 3.1
- MS SQL Server

5.3. Вимоги до апаратного забезпечення

- Комп’ютер на базі процесору Intel Core i3 і вище
- Оперативна пам’ять - не менше 2 Гбайт.

Пояснювальна записка
до дипломного проєкту
на тему: «Система електронного документообігу підприємства»

Київ – 2020 року

ЗМІСТ

ВСТУП

На сьогодні ми можемо спостерігати збільшення використання систем електронного документообігу. Не дивно, адже такі програмні продукти несуть багато користі сучасному бізнесу[4], а саме:

- Дозволяють оптимізувати бізнес-процеси;
- Пришвидшують діловодство на підприємстві;
- Дають пряму економію на накладних витратах (на папері, зносі принтерів, тощо);
- Збільшують надійність та безпеку документообігу (через неможливість втрати, пошкодження чи викрадення паперових носіїв).

Економічна вигода від використання систем електронного документообігу – неоціненна. Проте максимального ефекту можна досягти лише при повній відмові у користуванні паперовими документами. Таке можливо лише за умови, коли усі сторони, що взаємодіють на теренах українського економічного простору перейдуть на електронний документообіг. Проте багато підприємств наразі не можуть собі дозволити використовувати існуючі програмні продукти з тих чи інших причин.

Тож метою цієї роботи є розробка прототипу корпоративної системи електронного документообігу, що б могла задовольнити частину таких підприємств. Для цього необхідно проаналізувати існуюючий ринок систем електронного документообігу, знайти ключові недоліки, що заваджують у активному впровадженні цих продуктів та врахувати необхідність вирішення цих проблем в розробці свого додатку.

Досягнення поставлених завдань може дозволити збільшити відсоток використання систем електронного документообігу у бізнесі та пришвидшити цифрову трансформацію суспільства.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Система електронного документообігу (СЕД), або Системи Автоматизації Документообігу – це комп'ютерна автоматизована система, що використовується для ідентифікації, збереження, відстежування (трекінгу), та керуванням циркуляцією документів[7].

Невід'ємні правила будь-якої системи електронного документообігу:

- Єдина база документів;
- Однозначна та надійна ідентифікація документу;
- Наявність системи пошуку документів;
- Наявність системи контролю життєвого циклу документів.

1.1. Класифікація СЕД

Системи електронного документообігу можуть бути класифікованими за багатьма параметрами.

За функціональним призначенням:

- Електронні архіви. Системи електронного документообігу, що, переважно, орієнтовані на ефективне зберігання та пошук інформації. Вони мають добре розвинуті функціональні підсистеми пошуку та каталогізації[7]. Зазвичай відсутні або нерозвинуті системи циркулювання документів та маршрутизації
- Системи керування потоками документів. Системи, переважною функціональною орієнтацією яких є організація та контроль руху документів по заданих маршрутам. Основними модулями є підсистеми циркулюваання та маршрутизації документів. Поділяються на системи з жорсткою та гнучною маршрутизацією. Перші мають обумовлені на етапі розробки, чітко та жорстко задані етапи взаємодії з працівниками

та зовнішніми системами, процедури комунікації системи з операторами. Другі – покладають задання маршрутів документів на керівництво підприємства та мають можливість їх динамічної зміни.

- Гібридні системи. Такі системи є своєрідним поєднанням елементів всіх попередніх категорій. Містить деякі їх функції, переваги та недоліки у різних пропорціях.

За форматом постачання:

- Універсальні системи. Системи із стандартним набором функцій для всіх користувачів. Відсутність будь-якої кастомізації для користувачів. Найдешевші в використанні. Часто не можуть ефективно виконувати покладені на них функції, не повністю відповідають потребам та запитам конкретних споживачів.

- Індивідуальні рішення. Системи розроблені спеціально для окремих клієнтів, враховуючи всі їх потреби, запити та бізнес-процеси їх підприємств. Найдорожчі для розробки та впровадження у використання. Також, дуже високі часові та фінансові затрати на післяпродажну підтримку, так як, фактично, кожен споживач має свою, незалежну систему із своїми недоліками та багами, що потребують робочого часу окремих команд розробників для їх виправлення.

- Комбіновані системи. Поєднують елементи двох попередніх категорій: складаються з основної «базової» системи, та численних конфігурованих модулів (плагінів). Модулі часто розробляються спеціально для конкретних користувачів. Мають середню вартість та гнучкість, що цілком достатньо для задоволення потреб багатьох споживачів.

За предметною областю:

1. Універсальні;

2. Спеціалізовані:

- Виробничі;
- Технологічні;
- Кадрові;
- Складські;
- Архівні;
- Інші.

Також, виробники систем електронного документообігу часто використовують власні класифікації для різних лінійок своїх продуктів, переважно з маркетинговою метою.

1.2. Функціональність СЕД

Сучасні системи електронного документообігу дають своїм користувачам велику кількість можливих функцій:

- Зберігання документів. Базова функція всіх систем електронного документообігу. У різних виробників відрізняється способом та форматом збереження документів. Так, деякі системи зберігають документи як файли, у вигляді аттачменту до електронної картки документу, інші – як власне інформація всередині системи, у вигляді простого тексту, або розмітки певною мовою (html, xml), можливі комбінації підходів. Також відрізняються джерела зберігання; для файлового підходу: локальні дискові сховища, різні BLOB (Binary Large Object) сховища, хмарні диски та інші; для інформаційного підходу: реляційні СУБД різних виробників (MsSql, Oracle), NoSQL СУБД (MongoDB, Cassandra), а також багато інших способів[4].
- Пошук документів. В системах електронного документообігу існує декілька основних видів пошуку: простий пошук, логічний пошук,

повнотекстовий пошук. Простий пошук – пошук лише по полям (атрибутам) документів, характерний для систем керування потоками документів. Логічний пошук – пошук, що дозволяє формувати запити з використанням логічних операторів. Повнотекстовий пошук – пошук по внутрішньому тексту документів, використовується у деяких електронних архівах, дає найкращі пошукові можливості, потребує систем індексації змісту документів.

- Підтримка різних видів документів. У більшості систем керування потоками документів маршрути та дії над документами, доступні користувачам, залежать від типу документу.

- Збереження реквізитів. Метаінформація про документ, така як автор, адресат, дата створення, назва документа тощо називається реквізитами. Збереження реквізитів доступно в абсолютній більшості сучасних систем електронного документообігу, проте їх список різний, залежно від виробника. Часто доступний набір реквізитів залежить від типу документу.

- Маршрутизація документів. Система маршрутизації документів дозволяє встановлювати порядок взаємодії робітників з документами в системі. Ключова функція систем керування потоками документів.

- Синхронізація з іншими системами підприємства. Більшість систем автоматизації документообігу мають можливість імпорту та синхронізації інформації про підприємство (список працівників, структуру департаментів, посади, список адміністраторів тощо) з LDAP або бази даних підприємства.

- Версії документів. Версіонування документів дозволяє прослідкувати за ланцюгом змін та відкочувати їх при виявленні помилок, за прикладом систем контролю версій, що використовуються в розробці програмного забезпечення.

- Покращене архівування. Деякі системи пропонують можливість виділити окремі архівні сервери з низкою потужністю та вартістю обслуговування, на них зберігаються заархівовані документи, або ті, що рідко використовуються.

- Аудит дій. Запис усіх дій користувачів, таких як вхід в систему, перегляд та редактування документів та інше. Дозволяє розслідувати випадки недобросовісної поведінки та шахрайства з боку працівників. Як бонус – такий модуль значно полегшує відтворення багів для їх подальшого виправлення.

- Статистика. Модуль статистики по користуванню документами може допомагати у оцінюванні якості роботи працівників підприємства та збільшуванні їх ефективності.

- Автоматизація додавання нових документів. Деякі системи електронного документообігу пропонують можливість автоматичного додавання нових документів, наприклад отриманих електронною поштою.

- Можливість взаємодії з органами виконавчої влади. Для цього система повинна були ліцензована та перевірена на можливість інтеграції з Системою Електронної Взаємодії Органів Виконавчої Влади (СЕС ОВВ) Міністерством цифрової трансформації України.

- Можливості накладення резолюцій. Така функціональність важлива для систем керування потоками документів для можливості симулювати усі аспекти паперового документообігу.

- Конструктор документів. Деякі виробники пропонують можливість користувачам самим створювати власні види документів, конфігуруючи необхідні поля та реквізити.

1.3. Аналіз представлених на ринку рішень

Сучасний ринок програмного забезпечення для в Україні надзвичайно закритий. З історичних причин абсолютна більшість продуктів представлені на ньому пропрієтарні, що означає закритий вихідний код та повну відсутність будь-якої документації у вільному доступі. Це стосується й систем електронного документообігу. Більше того, навіть інформація про функціональність систем часто закрита: на сайтах багатьох виробників є лише рекламна інформація без переліку всіх реалізованих функцій системи[14]. Певну суттєву інформацію часто можна отримати лише після реєстрації з підтвердженням особистих даних, або з сторонніх неофіційних джерел. Іноді повну функціональність системи можна дізнатися лише в телефонній розмові з представником відділу продажів компанії виробника[14].

Системи електронного документообігу постійно оновлюються, для здобутку нових можливостей. Також, перед крупними виробниками стоїть проблема державного ліцензування систем, для можливості їх використання в органах державної влади, та інших установ з бюджетним фінансуванням, що змушує розробників постійно оновлювати системи відповідно до постійних змін у законодавстві. Інформація ж про продукт та його функціональність оновлюється не завжди оперативно. Крім того, ринок є досить динамічний, що характеризується високою частотою змін у ціноутворенні.

Тож, згідно вищеназваних причин, інформація, представлена в даному розділі (про функціональність та ціни) може бути неповною чи застарілою, незважаючи на всі спроби зібрати якомога актуальніші та точні дані.

1.3.1 FossDoc

FossDoc – система електронного документообігу українського виробництва, відноситься до класу систем керування потоками. Має усі необхідні для такої системи функції, такі як: зберігання документів, їх маршрутизація, історія версій, журнал тощо (залежить від комплектації)[4].

Переваги:

- Можливість попереднього перегляду документу;
- Інтеграція з поштою;
- Підтримує версіонування записів;
- Підтримує різні бази даних;
- Ліцензована та інтегрована з системами електронної взаємодії українських виконавчих органів влади (СЕС ООВ);
- Багато інформаційних матеріалів.

Недоліки:

- Відсутній повнотекстовий пошук;
- Пропрієтарність;
- Низька швидкодія (за відгуками користувачів)[11].

Щодо ціни на цю систему електронного документообігу, вона вираховується нелінійно та залежить від багатьох факторів, таких як доступні функції, кількість користувачів, обрана специфікація серверу, додаткові послуги тощо.

Ціна за 1 користувача змінюється нелінійно, змінюється залежно від кількості робітників, від 90 (при 6) до 52 (при 2000) доларів США за 1 місце[4].

Кожна функція, такі як: історія документів, журнал, накладення резолюцій, авторизація користувачів через домен windows коштує окрему ціну. Усього таких додаткових функцій 24. Ціна на кожну від 100 до 400 доларів США.

Також є вибір варіанту бази даних. Різні опції коштують від 300 (MySQL) до 1000 (Oracle) долларів США (це не включає вартість на ліцензію на базу даних та відповідний сервер, це лише вартість можливості їх використання).

Також пропонуються додаткові послуги по встановленню програмного забезпечення та навчанню користування системою. Ціна кожної – 1000 доларів США.

На Рис. 1.1 зображено скриншот цінового калькулятор FossDoc.

Отже мінімальна можлива ціна – 1037 доларів США, за версію на 6 користувачів з можливістю лише зберігати та переглядати документи, яка використовує базу даних MySQL. Повна версія, з усіма доступними функціями, максимальною кількістю користувачів та додатковими послугами коштує 110070 доларів США.

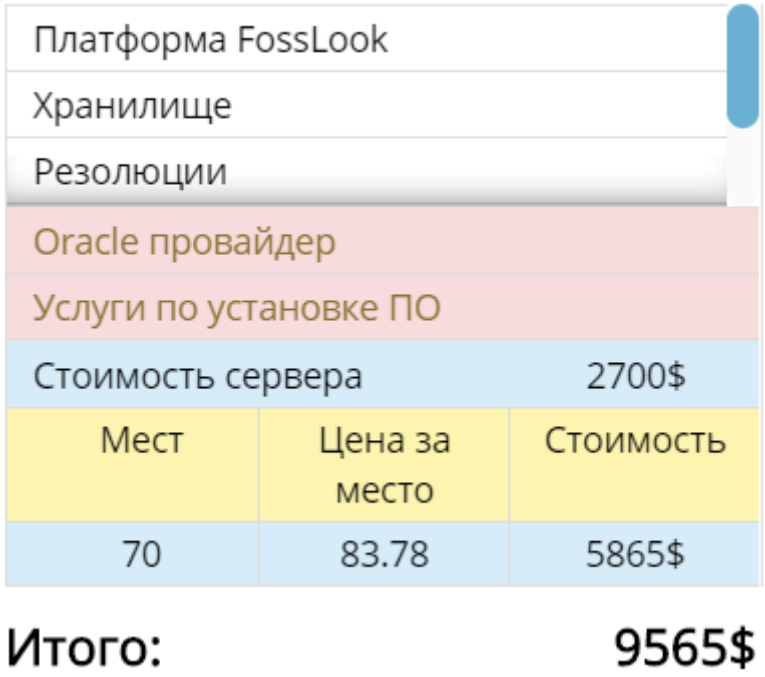


Рис. 1.1 Скриншот цінового калькулятора FossDoc

Також існує безкоштовна пробна версія з мінімальним набором доступної функціональності і обмеженням в 5 користувачів, проте для реального робочого використання така конфігурація підходить мало.

Післяпродажне обслуговування коштує 15% від ціни покупки програмного пакету в рік.

Необхідно зазначити ще є можливість користуватись системою без покупки, щомісячно сплачуючи підписку, проте ціна підписки на 3 роки дорівнює ціні покупки продукту, що використовується, тому така опція в довгочасній перспективі є невигідною, проте це можна використати для пробного запровадження електронного документообігу в малому підприємстві.

1.3.2 Docsvision Архів

Docsvision Архів – частина комплекту офісного програмного забезпечення компанії Docsvision, проте Архів – умовно незалежна система, що встановлюється самостійно та має окремі презентаційні матеріали, тому є сенс його окремого розгляду[7].

Архів являє собою стандартну систему електронного архіву та реалізує усю необхідну функціональність для програми такого типу. Особливістю є стійкість до високих навантажень системи.

Переваги:

- Можливість архівувати документи, переносячи їх на виділені бази даних на серверах з нижчою потужністю, що дозволяє заощаджувати на їх оренді та знизити навантаження на основний сервер;
- Гарна інтеграція з іншими програмними рішеннями екосистеми виробника;
- Висока швидкодія;
- Висока надійність.

Недоліки:

- Певна залежність від інших програм Docsvision;

- Характерна для архівів відсутність маршрутизації документів;
- Висока ціна;
- Пропрієтарність.

Ліцензія на 1 користувача коштує близько 110 доларів США за найдешевшим планом з обмеженням в 100 працівників, середній за ціною план – 180 доларів США за місце, без обмеження на кількість працівників, але з недоступними функціями для високонавантажених систем. Повна ліцензія коштує близько 260 доларів США за 1 працівника.

Щодо вартості самого програмного забезпечення – як і в попередній оглянутій системі вона сильно залежить від конкретної комплектації. Ціни залежать від виду ліцензійнго плану, що описані вище. Вони на порядок вище ніж в попередній системі та вимірюються у тисячах доларів США за функцію.

1.3.3 Docsvision Діловодство

Docsvision являє собою платформу з дуже широкими функціональними можливостями. Клієнт може сконфігурувати версію програмного забезпечення що може задовольнити майже будь які потреби. Також важливою особливістю є стійкість до високих навантажень системи[7].

Переваги:

- Широкі функціональні можливості;
- Висока надійність;
- Висока швидкодійність.

Недоліки

- Надзвичайно висока ціна;
- Неможливість взаємодії з органами державної влади в Україні ;
- Пропрієтарність.

Ціноутворення ліцензій на робочі місця повністю повторюють Docsvision Архів, власне це одні й ті ж ліцензії, заплативши за них можна використовувати будь-які рішення компанії Docsvision.

Порядок цін на окремі функції такий самий як в Docsvision Архів. Майже вся функціональність системи складається з додаткових платних модулів, навіть стійкість до високих навантажень системи, що є головною конкурентною перевагою, це додаткова опція, не включена в базовий набір. Ціна базової версії, без будь яких функціональних можливостей різна для різних видів ліцензійних планів – від 300 до 8800 доларів США. Список функцій, доступних для покупки також залежить від плану – так, наприклад, модуль повнотекстового пошуку, що коштує 2400 доларів США, доступний для вибору лише для найдорожчого ліцензійного плану (260 доларів США за працівника)[7].

Через дуже велику кількість додаткових модулів ціна версії з повними функціональними можливостями є надзвичайно високою. Необхідно зазначити, що такі можливості у абсолютній більшості випадків є надмірними, проте це не змінює факту дороговизни цієї системи.

1.3.4 Docassemble

Docassemble – одна з небагатьох відкритих рішень електронного документообігу. Вона являє собою не готову систему електронного документообігу, а платформу для створення власних систем на мові Python[10]. Тому не можна її напряму порівнювати з іншими системами напряму, за тими ж критеріями, проте за кордоном таке рішення популярне за кордоном і створює значну конкуренцію класичним консервативним виробникам. Функціональні можливості обмежені лише часом команди розробників, що відповідає за створення системи.

Переваги:

- Безкоштовність;
- Ліцензія MIT, що дозволяє модифікувати надану систему.

Недоліки:

- Не дає готового програмного рішення, тому абсолютно не підходе для пересічних малих підприємств.

1.3.5 E-Docs

E-Docs – система електронного документообігу українського виробника. Являє собою ціле сімейство різних типів та призначень. Має можливість взаємодії з виконавчими органами влади, більше того – система використовується в деяких державних та комунальних підприємствах[8].

Переваги:

- Широкі функціональні можливості;
- Сучасний інтерфейс;
- Інтеграція з Office Online;
- Ліцензована та інтегрована з системами електронної взаємодії українських виконавчих органів влади (СЕС ООВ).

Недоліки:

- Закритість інформації;
- Пропрієтарність.

На сайті виробника відсутня будь-яка інформація про вартість продукту та принципи ціноутворення. Проте, на державній платформі аукціонів для проведення тендерів «Prozorro» є лот на закупівлю комунальним підприємством «Київтеплоенерго» ліцензій користувачів для системи електронного документообігу E-Docs. Перемогла компанія-виробник запропонувавши ціну в 2160000 гривень[9]. Нажаль, про вартість самого програмного забезпечення, його склад та навіть кількості закуплених ліцензій інформація відсутня.

1.4 Спільні недоліки існуючих СЕД

З попереднього підрозділу можна зробити висновок, що ключовим недоліком сучасних популярних систем електронного документообігу є їх надзвичайно високі ціни. Для малих та середніх підприємств необхідність таких суттєвих фінансових витрат стає неподоланною перепоною на шляху до цифрової трансформації.

Для більш крупних підприємств важлива відкритість програмного коду та можливість його самостійної модифікації, для повної відповідності своїм потребам та інтеграції з іншими своїми програмними системами.

Фактично, крупні компанії мають дуже обмежені можливості впливу на виробництво готових програмних продуктів. Це змушує їх розробляти власні пропрієтарні рішення, що призводить до повної дестандартизації та неможливості міжпідприємчої та міжсистемної інтеграції такого програмного забезпечення.

Також, серед представлених рішень, жодна з систем потоків документів не має повнотекстового пошуку, або така можливість має великі обмеження, а електронні архіви не мають систем маршрутизації документів. На українському ринку програмного забезпечення фактично не представлено повноцінних систем гібридного типу.

Отже, оскільки метою розроблюваної системи є вирішення проблем та недоліків існуючих систем електронного документообігу, можна сформулювати список ключових вимог до створюваної в рамках цієї роботи системи електронного документообігу підприємства:

- Система повинна мати відкритий вихідний код та open-source ліцензію.
- Система повинна поширюватись безкоштовно, за принципом as-is.
- Базова версія системи повинна бути повністю придатна для використання в малих підприємствах та задовольняти їх базові потреби.

- До базової версії повинна бути включена система маршрутизації документів.
- До базової версії повинна бути включена можливість повнотекстового пошуку, що уже зможе вигідно виділити система на фоні існуючих.
- Система повинна бути гнучкою та добре придатною до модифікацій. Крупні підприємства повинні мати можливість будь-яких переробок системи під свої потреби та завдання та створення інтеграцій з іншими своїми системами, в тому числі пропрієтарними.

За умови задоволення цих вимог, система уже буде придатною для внутрішнього використання невеликими підприємствами і буде в цьому досить вдалою та конкурентоспроможною. Згодом, в подальшому розвитку можна буде додавати нові можливості, такі як:

- Міжфілійний зв’язок;
- Модуль розпізнавання тексту з фото та сканів вхідних паперових документі;
- Інтеграції з популярним офісним програмним забезпеченням;
- Використання електронного цифрового підпису;
- Ліцензування для можливості роботи з органами державної влади та на підприємствах з державною чи комунальною власністю.

ВИСНОВКИ ДО РОЗДІЛУ 1

В цьому розділі було розглянуте поняття систем електронного документообігу, їх ознаки та властивості, класифікацію, основні функціональні можливості.

Було проведено аналіз та порівняння програмних продуктів деяких представників українського ринку систем електронного документообігу, а саме:

- FossDoc;
- Docsvision Архів;
- Docsvision Діловодство;
- Docassemble;
- E-Docs.

Визначено основні спільні недоліки розглянутих систем, ними виявились висока ціна та закритість програмного коду. Встановлено основні функціональні потреби для конкурентоспроможної системи електронного документообігу. Сформульовано вимоги для розроблюваної системи:

- Безкоштовність;
- Відкритість;
- Відповідність до базових потреб підприємств;
- Легкість модифікації.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ

Завдання цього розділу – опис сучасних підходів та практик у розробці програмного забезпечення, їх порівняння та визначення оптимального рішення для розроблюваної системи. Також тут представлено детальний структурний опис реалізації обраної архітектури та використаних паттернів проектування.

2.1 Види архітектур програмного забезпечення

2.1.1 Цільна монолітна архітектура

Такий вид архітектури описує структуру програми яка являє собою одне ціле. Вона не передбачає поділення логіки програми на окремі абстрактні частини[2]. При цьому, такий підхід не заперечує поділ програми на структурні компоненти. Є найстарішою з нині існуючих архітектур.

Прикладом такої архітектури є MVC у класичному, не модифікованому варіанті. В MVC система складається з Моделей (model - M), що є описом сутностей даних, Представлень (view - V), що є описом користувацького інтерфейсу, Контроллерів (controller - C), що є ланкою зв'язку між моделями та представленнями. Логіка системи може повністю міститися в контроллерах, або ж частково передаватись до моделей, залежно від реалізації. Проте, такий поділ носить не логічний, а лише структурний характер.

Умовна схема цільної монолітної архітектури представлена на Рис 2.1

Переваги:

- Проста структура. Для розробки системи з цільною монолітною архітектурою треба мати мінімальні знання та навички;
- Швидкодійність (при правильній реалізації), через відсутність втрат на пересилання інформації;
- Неможливість проблеми "оверінжинірінгу", нагромадження зайвих абстрактних логічних шарів.

Недоліки:

- Нездатність системи до масштабування;
- Складність підтримки та обслуговування готової системи;
- Складність тестування;
- Складність в розширенні функціональності;
- Заплутаність, «непрозорість» бізнес-логіки.

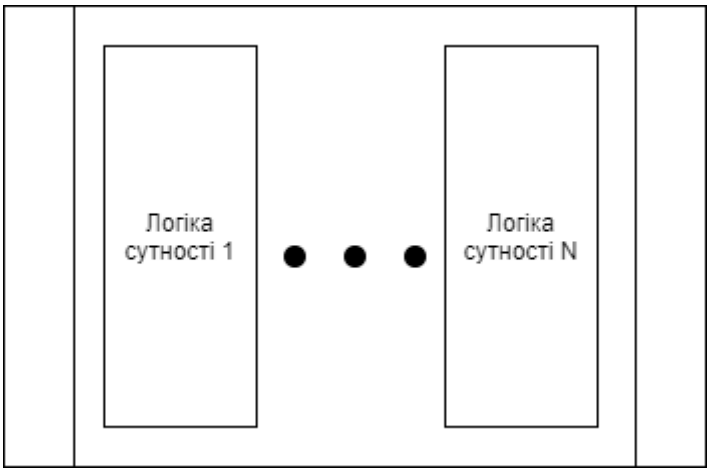


Рис. 01 Цільна монолітна архітектура

Наразі така архітектура є застарілою, та не використовується в реальних проектах. Можливе застосування з навчальною метою, або в малих програмах, де будь-яке ускладнення архітектури є недоцільним.

2.1.2 Багатошарова монолітна архітектура

Багатошарова архітектура є однією з найбільш використовуваних. Вона складається з шарів, що інкапсулюють логіку різних рівнів абстракції. Кожен шар незалежний та зв’язаний лише з двома іншими, вище та нижче на 1 рівень абстракції. Запити проходять ланцюжком через всі рівні [2].

Багатошарова архітектура є однією з найбільш використовуваних. Вона складається з шарів, що інкапсулюють логіку різних рівнів абстракції. Кожен шар незалежний та зв’язаний лише з двома іншими, вище та нижче на 1 рівень абстракції. Запити проходять ланцюжком через всі рівні.

Класичним прикладом є модель OSI.

Умовна схема багатошарової архітектури представлена на Рис 2.2

Переваги:

- Незалежність шарів. Абстрактні логічні шари ізольовані. Будь яка модифікація логіки одного шару, при збереженні зовнішнього інтерфейсу, не впливає на роботу інших.
- Простота підтримки.
- Простота тестування.
- Надійність.
- Можливість розширення.
- Відносна простота розробки.
- Швидкодія (при правильній реалізації). Втрати на переміщення даних між шарами не значні, оскільки відбуваються на одній машині.
- Чітка та логічна структура програми.

Недоліки:

- Часто виникає проблема “оверінжинірінгу”, нагромадження зайвих абстрактних логічних шарів. В такому випадку кодова база стає громіздкою та знижується швидкодія, через марне пересилання даних через багато шарів без їх фактичної зміни.
- При додаванні нових сутностей необхідно розширювати усі шари.
- Обмежені можливості швидкого масштабування, оскільки система працює на 1 машині.

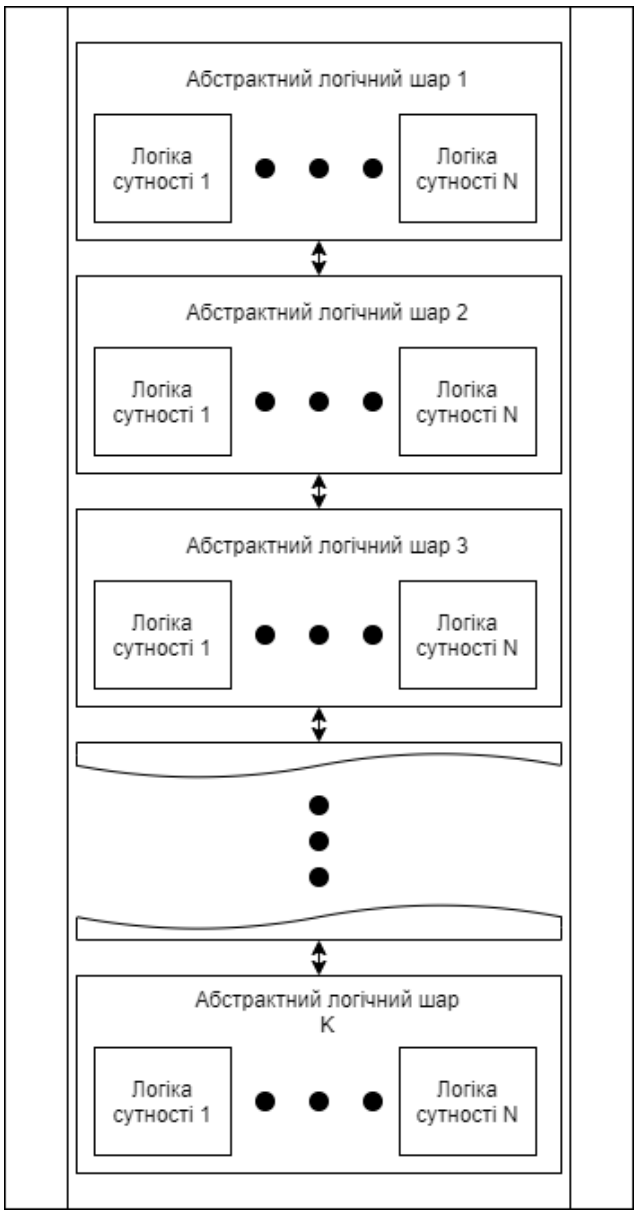


Рис. 02 Багатошарова архітектура

Дана архітектура широко використовується у сучасній розробці програмного забезпечення. В сфері корпоративного програмного забезпечення - де-факто стандарт, через надійність та простоту підтримки. Також широко використовується в проектах, ключовим критерієм для яких є швидкість розробки, для можливості першим зайняти ринок.

2.1.3 Сервіс-орієнтована архітектура

Сервіс-орієнтована архітектура передбачає поділення логіки на незалежні частини – сервіси. Сервіси інкапсулюють в собі самодостатні частини логіки. Основними характеристиками є низький рівень зв’язаності сервісів та

здатність до легкої зміни конкретної реалізації через стандартизацію інтерфейсів. Передбачається використання одного сервісу у декількох програмах[3].

Умовна схема сервіс-орієнтованої архітектури представлена на Рис 2.3

Переваги:

- Повторне використання кодової бази. Сервіс, написаний 1 раз може використовуватись у різних проектах.
- Легкість підтримки.
- Легкість модифікації. Незалежність сервісів дає можливість змінювати їх без небажаного впливу на інші компоненти системи.
- Простота тестування.
- Дуже висока надійність.
- Можливість розробки незалежними командами.

Недоліки:

- Складність архітектури.
- Низька швидкодія.
- Складність початку розробки. Сервіс-орієнтована архітектура передбачає наявність підсистеми зв’язку між сервісами, розробка та підтримка якої складна та дорога. І хоч подальша бізнес-логіка в сервісах у таких системах розробляється надзвичайно легко, це не може компенсувати затрати на розробку першого прототипу.

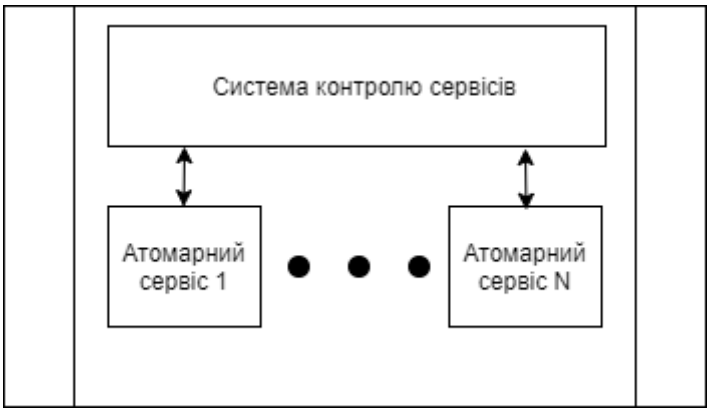


Рис. 03 Сервіс-орієнтована архітектура

Сервіс-орієнтовану архітектуру використовують у системах де насамперед важлива надійність, наприклад в банках та інших фінансових системах.

2.1.4 Мікросервісна архітектура

Мікросервісна архітектура описує програмну систему, що складається з багатьох невеликих окремих програм – мікросервісів. Кожен мікросервіс виконує одну, конкретну, незалежну функцію. Основним засобом міжсервісної комунікації є REST API. Різні компоненти розгортаються незалежно, часто на різних машинах. Також мікросервіси в одному проекті можуть бути написані на різних мовах програмування[3].

Умовна схема мікросервісної архітектури представлена на Рис 2.4

Переваги:

- Гнучкість. Через незалежність мікросервісів можна змінювати їх не впливаючи на роботу інших елементів системи.
- Висока здатність до масштабування. Різні частини додатку можуть бути розвернуті незалежно на різних серверах дуже швидко.
- Простота тестування.
- Можливість паралельної незалежної розробки багатьма командами. Використання REST API для зв’язку компонентів дозволяє розробляти

їх абсолютно незалежно, можливо навіть на іншому технологічному стеку або мові програмування.

Недоліки:

- Складність архітектури.
- Низька швидкодія. Через використання REST API для зв’язку між численними компонентами виникає велика кількість мережевих затримок при кожному запиті.
- Складне відтворення та виправлення багів. Через незалежне та динамічне розміщення компонентів, часто неможливо точно повторити середовище для відтворення багу. Також їх виправленню заважає відсутність централізованих логів.
- Низька надійність. Через незалежне розміщення компонентів на багатьох машинах виникає велика кількість точок відмови.
- Низька безпека. Пересилання усіх міжсервісних даних через REST API знижує загальну безпеку системи.

Рис. 04 Мікросервісна архітектура

Мікросервісний підхід до архітектури широко використовується в сферах де дуже важливі гнучкість та здатність до масштабування. Наприклад, в стартапах, медіадодатках та проектах з високим навантаженням.

2.1.5 Безсерверна архітектура

Безсерверна архітектура передбачає передачу керування машинними ресурсами на аутсорс, зовнішній компанії. Від розробника необхідно лише написати логіку, у вигляді окремих лямбда функцій, що розміщуються та виконуються в зовнішніх хмарних обчислювальних центрах. Таку інфраструктуру називають “функція як послуга” (FaaS - Function as a Service). Прикладами платформ для розробки додатків на основі безсерверної архітектури є Microsoft Azure Functions, Amazon Web Services Lamda, Google Cloud Functions[3].

Умовна схема безсерверної архітектури представлена на Рис 2.5

Структура проекту в безсерверній архітектурі багато в чому схожа на таку в Мікросервісній архітектурі, проте тут відсутній код, що відповідає за авторизацію, обробку http запитів тощо.

Переваги та недоліки представленої архітектури відносно інших повністю включають в себе такі в Мікросервісній архітектурі, тому є сенс розглянути переваги та недоліки лише відносно неї.

Переваги:

- Ціна та швидкість розробки.
- Автоматичне масштабування.

Недоліки:

- Низька конфіденційність. Дані необхідно зберігати на обладнанні сторонніх компаній.
- Через автоматичне масштабування швидкодія безсерверних систем навіть нижча ніж у мікросервісів.

- Ціна оренди серверних потужностей може різко та несподівано зростати при великому напливі користувачів, що іноді призводило навіть до банкрутства компанії власників програмних систем.
- Залежність від сторонніх компаній. Постачальник обраної хмарної платформи будь-якої миті може перестати надавати такі послуги, з тих чи інших причин. В такому випадку доведеться повністю переписувати систему на платформу іншого виробника чи на іншу архітектуру.

Рис. 05 Безсерверна архітектура

Наразі, безсерверна архітектура підходить для систем, де ключовою потребою є швидкість розробки. Така архітектура малопопулярна, в основному її використовують стартапи для швидкого виходу на ринок та перевірки попиту користувачів на ту чи іншу ідею проекту. Проте часто програми з іншими архітектурами частково використовують FaaS для окремих модулів системи.

2.2 Вибір оптимальної архітектури для системи електронного документообігу підприємства

Система електронного документообігу – корпоративний додаток, що визначає певні критерії вибору архітектури.

Система повинна бути:

- Конфіденційною. В документах часто можуть міститись дані клієнтів підприємства, які часто за договором не можна передавати третім особам.
- Безпечною. Попадання внутрішніх даних у вільний доступ може мати згубні наслідки.
- Надійною. Будь які простої в роботі через непрацюючу систему електронного документообігу це прямі фінансові втрати для підприємства.
- Швидкодійною. Низькі затримки в роботі програми підвищують комфорт роботи працівників.
- Гнучкою. Додавання нової функціональності системи, відповідно до потреб підприємства, повинно бути можливим.

Ці потреби відсортовано в порядку важливості для системи.

Водночас, для такого типу систем неважливі:

- Масштабування. Корпоративні системи не передбачають різкого непередбачуваного росту користувачів, тому можна точно розрахувати необхідні машинні потужності до початку використання.
- Можливість розробки незалежними командами. Така потреба виникає у крупних технологічних компаній в проектах з щоденними оновленнями, на які необхідно витрачати багато людських ресурсів. Корпоративні системи електронного документообігу до таких не відноситься.

Отже для розроблюваної системи однозначно не підходить безсерверна архітектура, адже вона не задовольняє головну вимогу – конфіденційність. Усі корпоративні системи повинні розміщуватись «On-premise» - на власному обладнанні.

В Мікросервісній архітектурі основні переваги, такі як масштабування та можливість паралельної розробки неважливі для розроблюваної системи, в той час як такі її потреби як надійність та швидкодійність не задоволені належним чином. Тож мікросевісна архітектура також не підходить.

Щодо цільного моноліту, така архітектура наразі є застарілою та має велику кількість вразливих недоліків, таких як: складне тестування, що призводить до проблем з надійністю, та відсутність будь-якої гнучкості. Отже цільний моноліт також не найкращий вибір для такої системи.

Тож лишаються багатошарова та сервіс-орієнтована архітектури. Перша – універсальне рішення. Друга краще підходе для систем такої предметної області, де важливість надійності значно переважає швидкодійність, наприклад фінансова сфера.

Оскільки система, що розроблюється у рамках цієї роботи, є системою загального призначення, то оптимальним вибором буде багатошарова архітектура. Також, для зниження зв’язності компонентів, можна додати елементи сервіс-орієнтованої, у вигляді організації деяких шарів як незалежних сервісів зв’язаних через впровадження залежностей (DI – Dependency Injection).

Така архітектура найкраще підходить для системи електронного документообігу підприємства і є найпоширенішим рішенням у сфері корпоративного програмного забезпечення[2].

2.3 Детальний опис обраної архітектури

2.3.1 Загальний опис 3-шарової серверної архітектури

Для розроблюваної системи було обрано багатошарову архітектуру.

Класичним її варіантом є 3-шарова (3-layer) архітектура. Її часто плутають з 3-рівневою (3-level) архітектурою. Рівень - це поняття фізичного розміщення додатку, у класичному розумінні будь який веб-додаток 3-рівневий, оскільки складається з бази даних, серверної частини та фронт-енду, а усі вони мають

різне фізичне розташування. Натомість 3-шарова архітектура – це саме логічний поділ та структура серверної частини програмної системи.

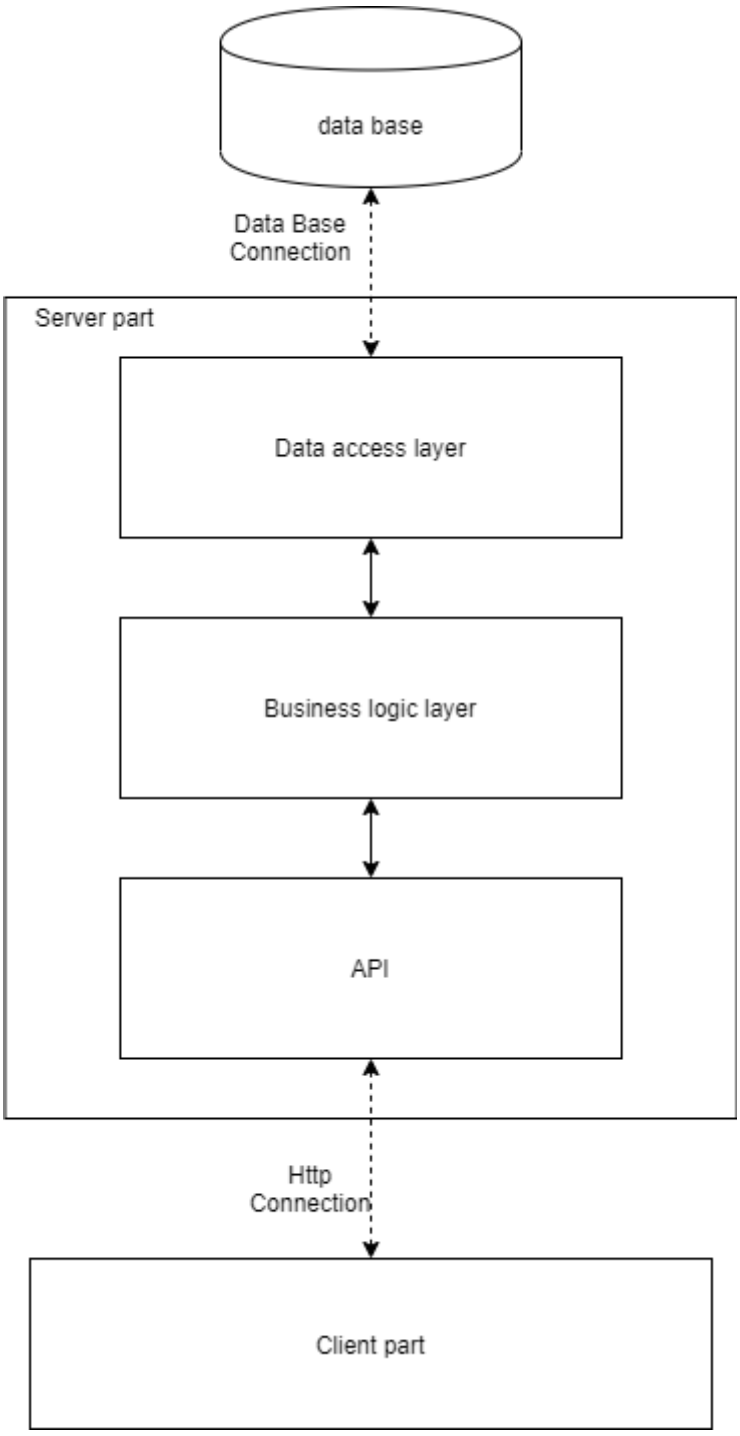


Рис. 2.6 3-рівнева архітектура

Отже, вона складається з таких шарів (layers):

- Шар доступу до даних (DA - Data Access layer)
- Шар бізнес-логіки (BL – Business Logic layer)

- Шар API (API layer)

Схема 3-рівневої архітектури представлена на Рис 2.6

2.3.2 Шар доступу до даних

Шар доступу до даних інкапсулює логіку роботи з базою даних, даючи нижчим за ієрархією шарам інтерфейс для їх пошуку, зміни, додавання та видалення. В цьому шарі реалізуються паттерни репозиторій (Repository) та Unit Of Work.

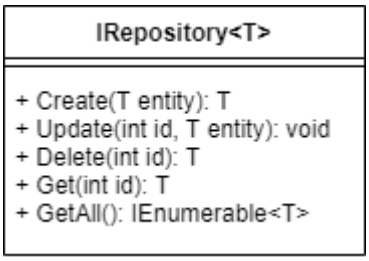


Рис. 2.7 Інтерфейс generic-репозиторію

Рерозиторій являє собою абстракцією над системою зберігання. Він інкапсулює доступ до конкретних джерел даних, запити до них формуються в його методах[5]. Загалом, репозиторії специфічні для сутностей, проте може бути generic-репозиторій з базовою CRUD (Create, Read, Update, Delete) логікою. Інтерфейс generic-репозиторію зображено на Рис. 2.7.

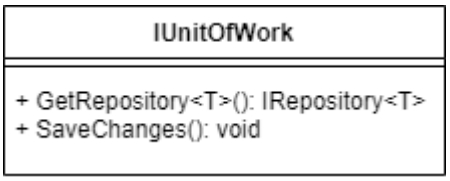


Рис. 2.8 Інтерфейс Unit of Work

Unit Of Work – фабрика репозиторіїв. Вона відстежує та затверджує зміну даних та дає інтерфейс доступу до специфічних репозиторіїв[5]. Інтерфейс Unit of Work зображено на Рис. 2.8. Її використання не обов’язкове для користування паттерном Репозиторій, проте вона дозволяє робити багато запитів до бази даних єдиною транзакцією та значно зменшує зв’язність, полегшуючи подальшу розробку системи. Структурна схема взаємодії Unit of Work та Репозиторіїв зображена на Рис. 2.9

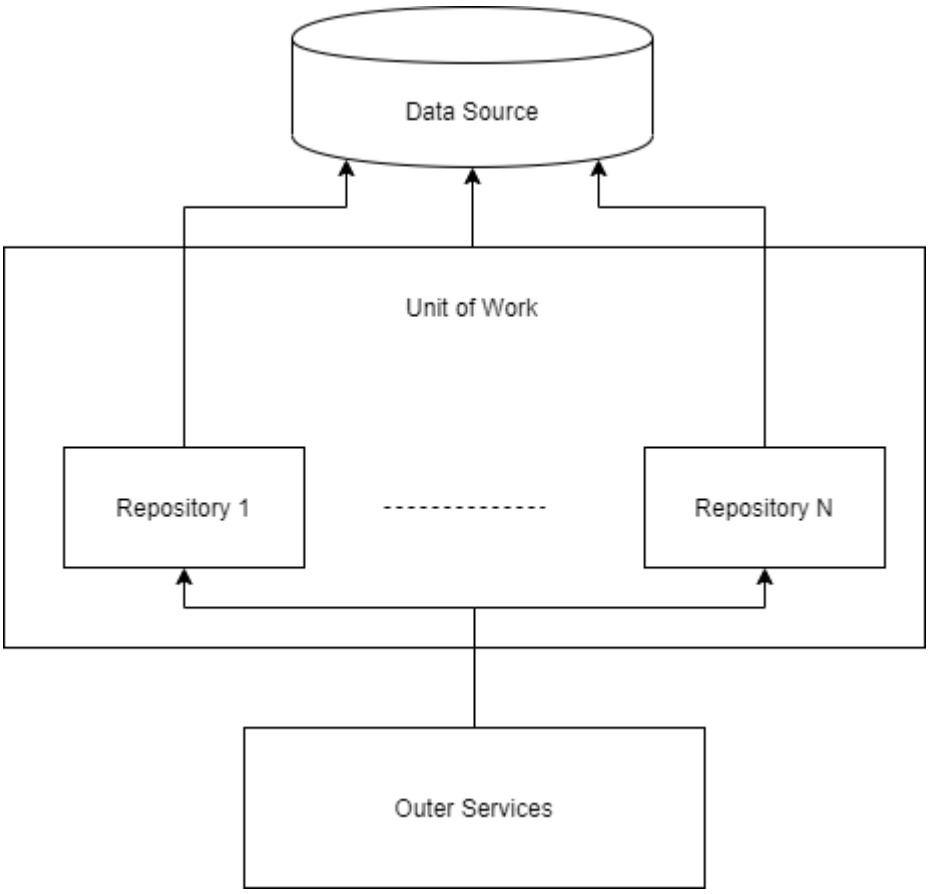


Рис. 2.9 Unit of Work та Репозиторії

2.3.3 Шар бізнес логіки

В шарі бізнес логіки знаходиться власне реалізація необхідної функціональності системи. Детальну структурну схему шару зображено на Рис. 2.10. Доступ до даних відбувається через шар вище за ієрархією. Для видачі результату роботи на шар нижче необхідно привести дані до відповідного вигляду – змапити до DTO.

DTO (Data Transfer Object) – модель сутності даних, придатна для серіалізації та передачі на клієнтську частину. Не має будь-якої логіки – лише інформаційний об’єкт. Може мати відповідний «аналог» з моделей сутностей бази даних, частково переймаючи його властивості, проте може бути комбінацією таких моделей бази даних або й цілком новою сутністю.

Маппінг – ручне приведення об’єкту одного типу до іншого, з копіюванням відповідних потрібних властивостей стандартних типів та рекурсивного маппінгу властивостей інших типів.

Часто маппінг до dto відбувається не в шарі бізнес логіки, виноситься на окремий, проте при використанні спеціальних бібліотек для автоматичного маппінгу за встановленими у вигляді виразів правилами таке розділення є надмірним.

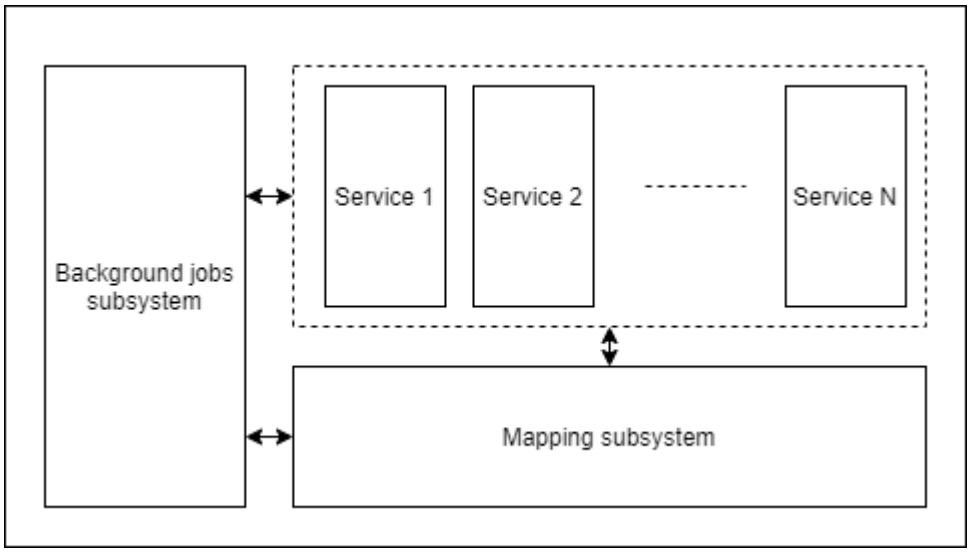


Рис. 2.10 Шар бізнес логіки

Вдалим рішенням є розбиття усієї логіки на окремі сервіси, специфічні для сутностей. Такі сервіси повинні створюватися через механізм DI (Dependency Injection) контейнерів. По аналогії з репозиторіями, сервіси в основному специфічні для сутностей з існуванням дженерік–сервісу з базовою CRUD логікою. Приклад такого generic-сервісу зображено на Рис. 2.11.

Dependency Injection – концепція передачі керування над створенням та ініалізації об’єктів програмі. Надбудови, що реалізують механізм Dependency Injection, які зазвичай є частиною фреймворку, називають DI контейнерами. Впровадження залежностей одна з форм інверсії керування (IoC – Inversion of Control) – передачі програмі контролю над кодом, його запуском та

виконанням, що призводить до значного зниження зв’язаності та збільшення гнучкості системи.

Також, частиною шару бізнес логіки є система планування завдань. Вона дозволяє запускати код з відтермінуванням чи за розкладом, через неї реалізуються періодичні завдання, такі як синхронізація даних, індексація тощо. Іноді це виносять в окремий шар, проте таке архітектурне рішення виправдане лише за умови достатньо великої кількості таких періодичних завдань.

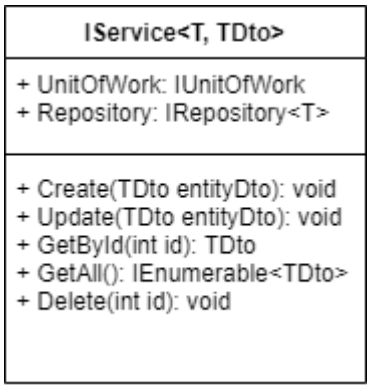


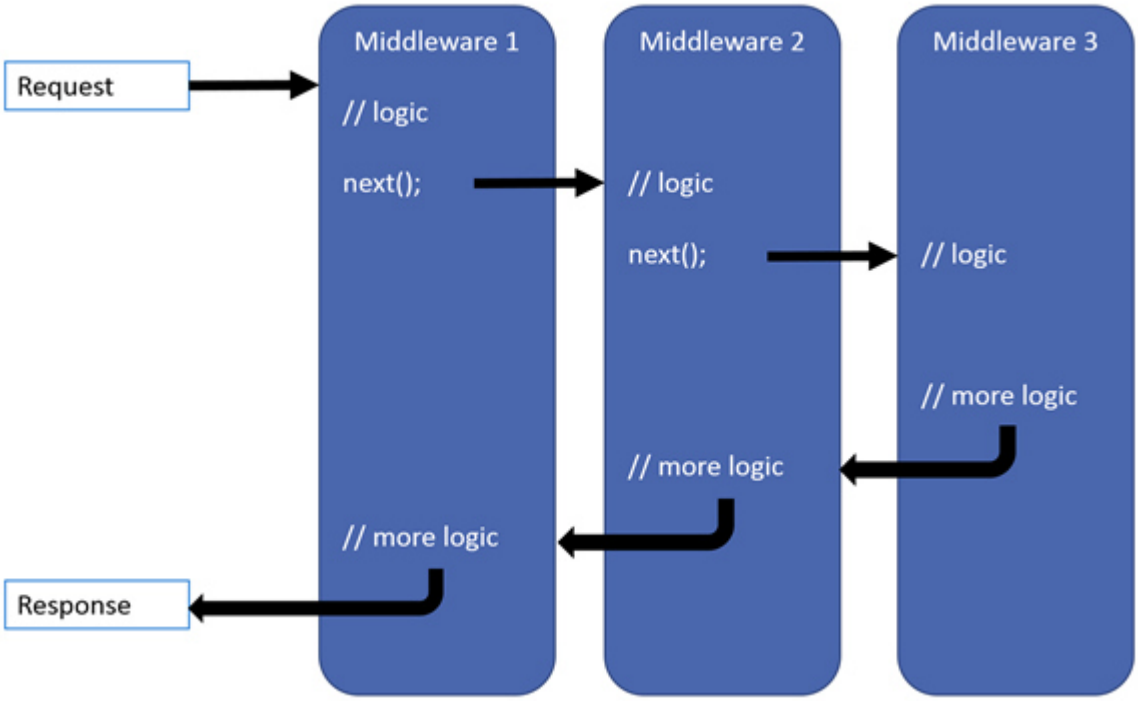
Рис. 2.11 Generic сервіс

2.3.4 Шар API

В шарі API відповідно розміщені контроллери, що описують кінцеві точки (endpoints) REST API.

Кожен публічний метод відповідного контроллера описує одну кінцеву точку. Для перетворення даних, переданих через http використовуються сервіси з шару вище за ієрархією. Також тут знаходиться логіка роботи зі статус-кодами. Часто контроллери реалізують специфічними для сутностей, дійсно, в такому випадку кожному контроллеру співставляється 1 сервіс, що є зручним для розробки, проте такий підхід не є єдиним та абсолютно правильним через можливі подальші суперечності у розміщенні тієї чи іншої кінцевої точки. Краще створювати контроллери за функціональним призначенням.

Власне шар API є вхідною точкою серверної частини додатку, тут формується конвеєр обробки http реквестів та резпонзів за допомогою компонентів middleware, схема роботи яких зображена на Рис 2.12.



2.12 Схема роботи компонентів middleware[12]

Middleware – компоненти, що вбудовуються в конвеєр обробки http (http pipeline) формуючи логічний ланцюг. Дані передаються з компоненту в компонент послідовно під час запиту, і в зворотньому порядку під час відповіді. [12] За допомогою кастомних middleware реалізуються такі функції як: авторизація та аутентифікація користувачів, первинна валідація dto, тощо.

Оскільки шар API є вхідною точкою усього додатку, то в ньому відбувається початкова ініалізація. Тут описано правила впровадження залежностей, імпортуються файли конфігурацій, впроваджуються деякі провайдери даних.

2.4 Архітектура фронт-енд частини

Клієнтська архітектура в основному залежить від технологічного стеку, на якому реалізована, тож будь який опис такої архітектури буде фактичним описом роботи відповідного використаного фронт-енд фреймворку. Щодо даної системи, тут використано Angular 8. Це передбачає поділення клієнтського коду на модулі за функціональним принципом – окремі функціональні блоки знаходяться в різних модулях. Структурна схема такого модулю представлена на Рис 2.13.

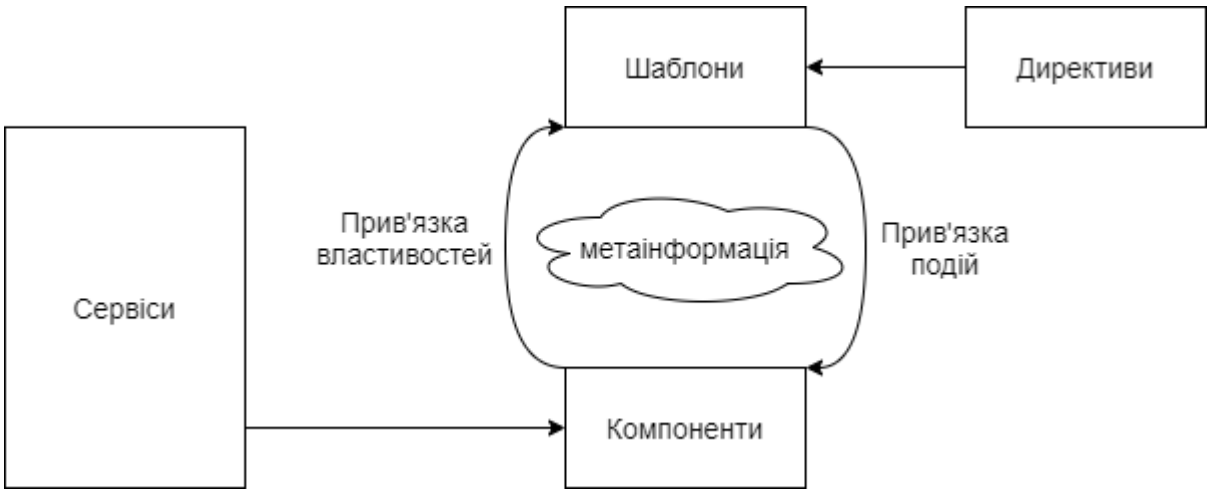


Рис. 2.13 Схема модулю клієнтської частини додатку

Спільна логіка та взаємодія з http знаходиться в сервісах, що впроваджуються через механізм dependency injection. Необхідна функціональність реалізується в компонентах. Вони складаються з логічних файлів та файлів розмітки і стилів. В перших, відповідно, описана логіка, в других – розмітка та верстка веб-сторінки. Вони пов’язані через механізм 2-сторонньої прив’язки даних[13]. Він передбачає при модифікації змінної в одному файлі – автоматична зміні в пов’язаному. Також існують директиви – програмні одиниці, що використовуються для зміни окремих елементів DOM дерева. Компоненти є окремими випадками директив.

ВИСНОВКИ ДО РОЗДІЛУ 2

В цьому розділі було оглянуто усі основні сучасні архітектури програмного забезпечення, розглянуто їх особливості, переваги та недоліки.

Проведено аналіз архітектурних потреб корпоративного програмного забезпечення в цілому та систем електронного документообігу підприємств зокрема. В результаті цього аналізу було встановлено, що оптимальним вибором для системи, що розроблюється є багатошарова архітектура.

Також, було проведено детальний огляд обраної архітектури, її структурної та логічної складових.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ

Завдання цього розділу – вибір та конкретного технологічно стеку, детальний опис деталей реалізації та обґрунтування тих чи інших рішень, зроблених в процесі розробки, а також основні сценарії використання системи.

3.1 Використаний технологічний стек

Ключовим фактором при виборі конкретних технологій у реалізації розроблюваної системи є робочий інтерес автора. Незважаючи на це, усі технології, які було вирішено використати в проекті, є найкращим або рівноцінним вибором серед конкурентів.

Серверна частина додатку написана на фреймворку ASP.NET Core 3 та мові програмування C#.

Мова C# - сучасна об’єктно-орієнтована мова програмування зі статичною типізацією, що стрімко розвивається, широко використовується у сучасній IT індустрії, має більше модернових синтаксичних конструкцій та можливостей ніж прямі конкуренти[6]. Гарний вибір для розробки корпоративних систем.

ASP.NET Core – сучасний веб-фреймворк в середовищі .NET Core. Надійний, швидкодійний, кросплатформний, має велику кількість бібліотек для різних потреб. Поширене рішення для розробки систем такого типу[1].

База даних – MS SQL Server. Не має суттєвих відмінностей від конкурентів для системи масштабу розроблюваної, проте є добре сумісною з обраною платформою, так як має того ж самого виробника – компанію Майкрософт[5].

Для зв’язку з базою даних використано ОРМ EntityCore Framework. Це майже безальтернативне рішення для систем на технологічному стеку Майкрософта та одна з найкращих ОРМ взагалі.

Впровадження залежностей відбувається за допомогою стандартного, вбудованого в ASP.NET Core DI контейнера.

Для маппінгу в dto використано бібліотеку Automapper, що дозволяє автоматично мапити різні об’єкти за раніше написаними правилами.

Бібліотека Hangfire використовується для реалізації системи запуску задач за розкладом чи з затримкою. Також вона надає можливість моніторингу за завданнями через візуальний інтерфейс[15].

Для індексації документів та повнотекстового пошуку використовується ElasticSearch. Це – одна з найпопулярніших платформ для реалізації високопродуктивного повнотекстового пошуку.

Бібліотеку EPPlus використано для формування файлів звітів (тобто файлів екселю).

Також використано Swagger – систему опису та автоматичного документування Rest API. Також Swagger надає Swagger UI tool – зручний візуальний інтерфейс для перевірки та ручного запуску кінцевих точок створеного REST API[16]. Дуже часто використовується у веб-додатках незалежно від предметної області.

Для клієнтської частини використано фронт-енд фреймворк Angular 8 та мову програмування TypeScript.

Angular 8 – один з сучаних популярних фронт-енд фрейворків. Серед конкурентів його вигідно вирізняє використання мови програмування TypeScript, та структуру додатку, дещо схожу на серверну архітектуру – з сервісами та впровадженням залежностей[1]. Це дозволяє розробляти додатки з низькою зв’язністю та логічною структурою.

Щодо TypeScript, це сучасна мова програмування, розроблена компанією Майкрософт. Фактично являє собою JS із строгою статичною типізацією, що значно полегшує розробку крупних додатків та є неоціненною перевагу для людей, що використовують класичні ООП мови на бек-енді. Транслюється компілятором в звичайний JavaScript.

Для створення розмітки веб-сторінок використано UI фреймворк Bootstrap 4, він значно полегшує верстку, в порівнянні з простим HTML та CSS.

3.2 Реалізація серверної частини додатку

Серверна частина являє собою Solution з декількома проектами. Основний - ASP.NET Core Web Api проект, який описує шар API та містить точку входу. Інші – бібліотеки класів. Структуру файлів Solution'у зображено на Рис. 3.1.

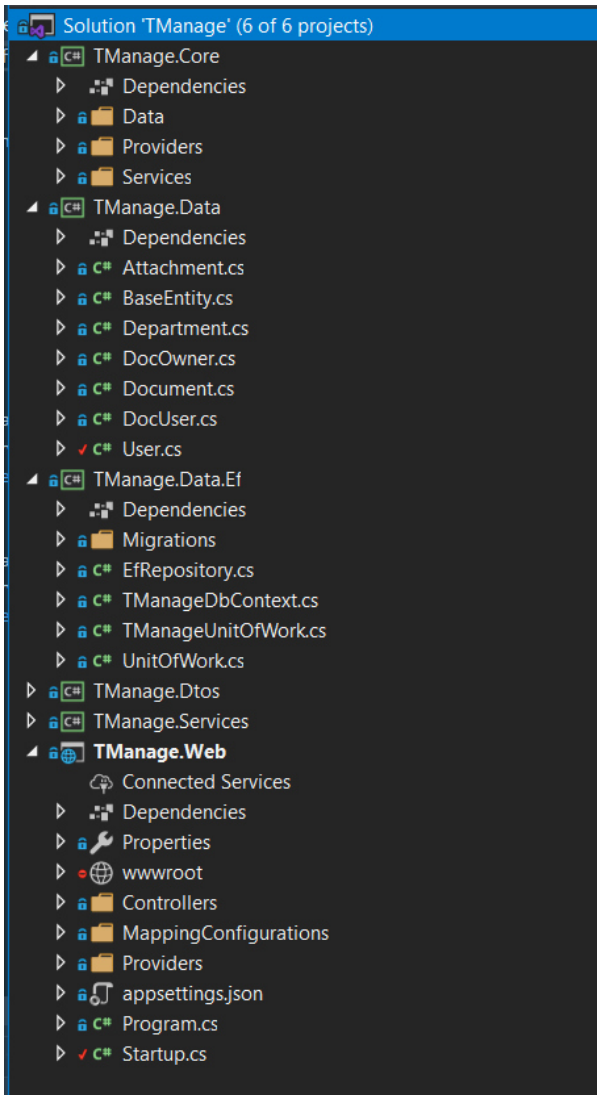


Рис 3.1 Файлова структура

3.2.1 Реалізація шару доступу до даних

В шарові доступу до даних реалізовано паттерни Repository та UnitOfWork. Також він включає опис контексту та моделей даних. Схему класів зображено на Рис 3.2. Для створення схеми бази даних використано підхід Code-First

ОРМ-системи Entity Core Framework. Вона передбачає автоматичну генерацію таблиць і ключів бази даних з моделей сутностей[5].

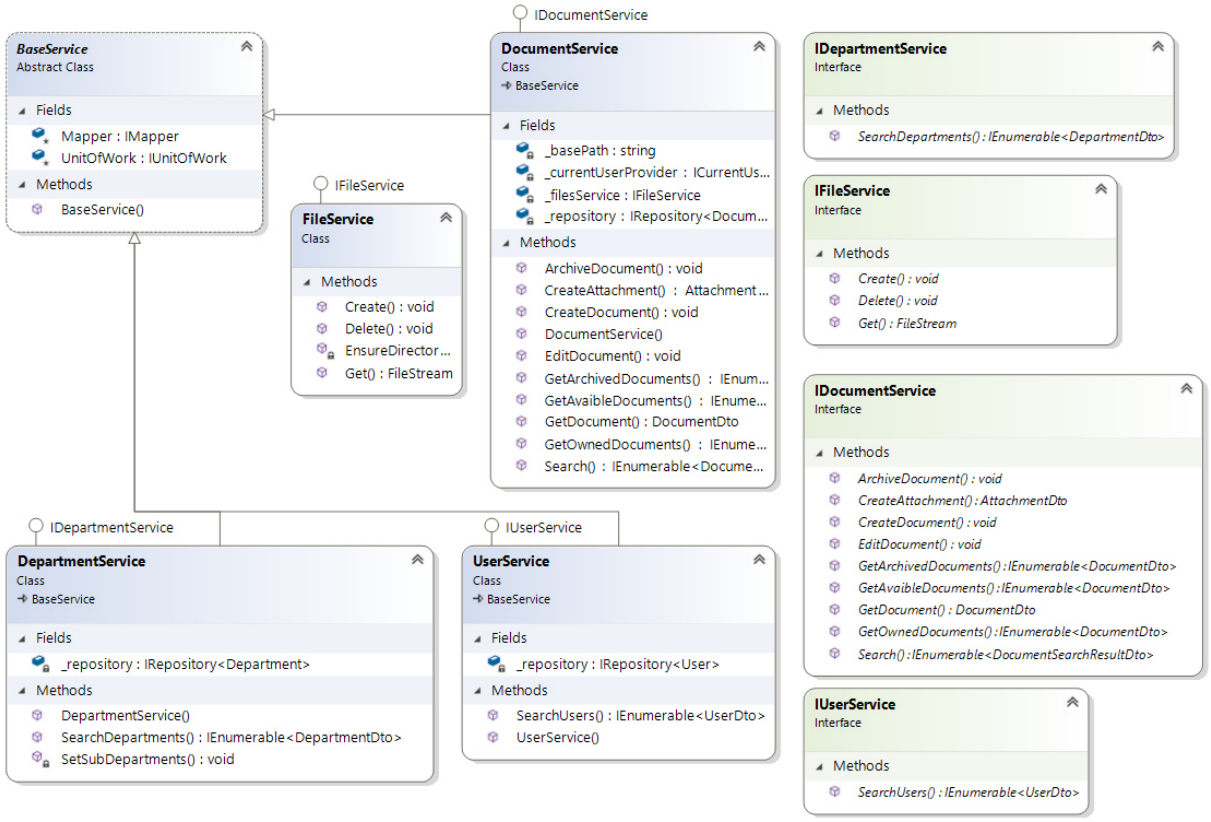


Рис 3.2 Схема шару доступу до даних

Для уточнення деяких зв’язків використано fluent api – спеціальний механізм Entity Core Framework’у для задання зв’язків та відношень між сутностями даних за допомогою “виразів”. Схему бази даних зображено на Рис. 3.3.

3.2.2 Реалізація шару бізнес-логіки

Шар бізнес-логіки складається з незалежних сервісів, власне у яких міститься логіка, та їх інтерфейсів. Схему шару зображено на Рис. 3.4.

Сервіси містять реалізацію логічно розділених частин функціональності додатку. Зазвичай вони є сутнісно-специфічними, тобто кожен сервіс відповідає за логічні операції над окремою сутністю бази даних. Проте це не є

обов’язковим, так, наприклад, сервіс FileService описує логіку збереження файлів на сховище та не має відповідної сутності бази даних.

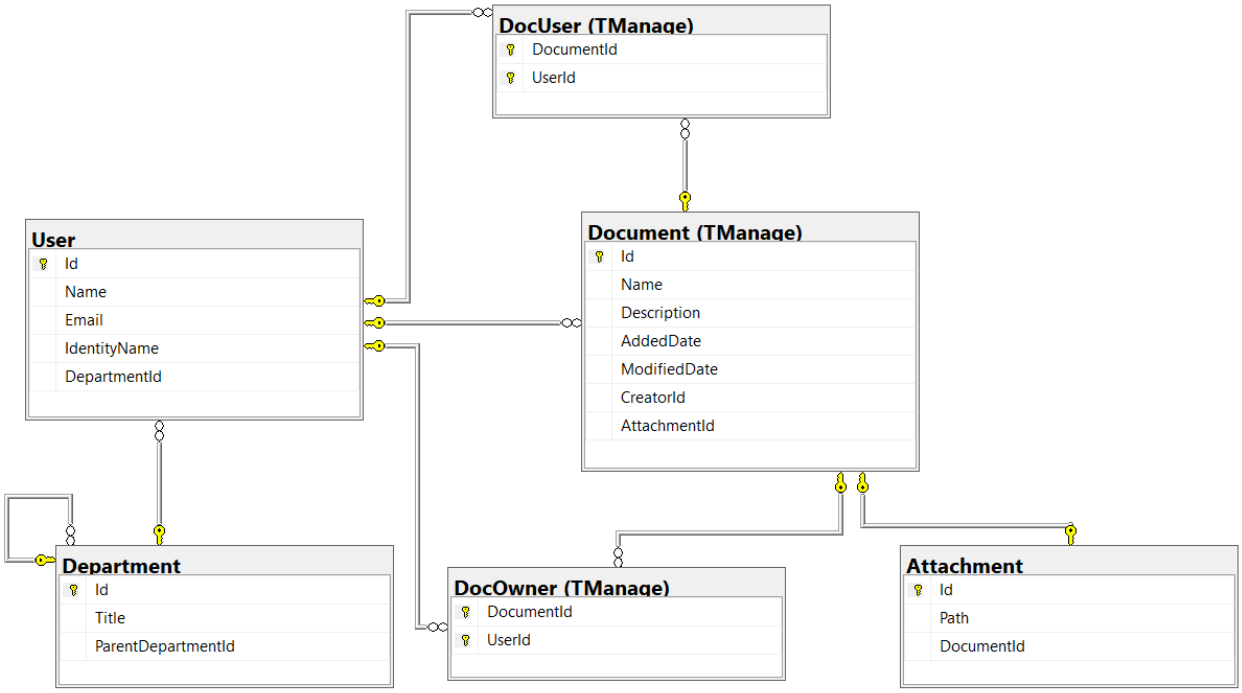


Рис. 3.3 Схема бази даних

3.2.3 Реалізація шару Web API

WEB API реалізоване в контроллерах основного шару серверної частини додатку. Схему шару WebApi зображено на Рис. 3.5.

Кожен публічний метод контроллера являє собою одну кінцеву точку. Вони згруповані за логічним принципом. Загалом є 3 контроллери:

- DocumentController, який містить в собі дії з документами
- UserController, який містить в собі дії з користувачами
- DepartmentController, який містить в собі дії з підрозділами (департаментами)

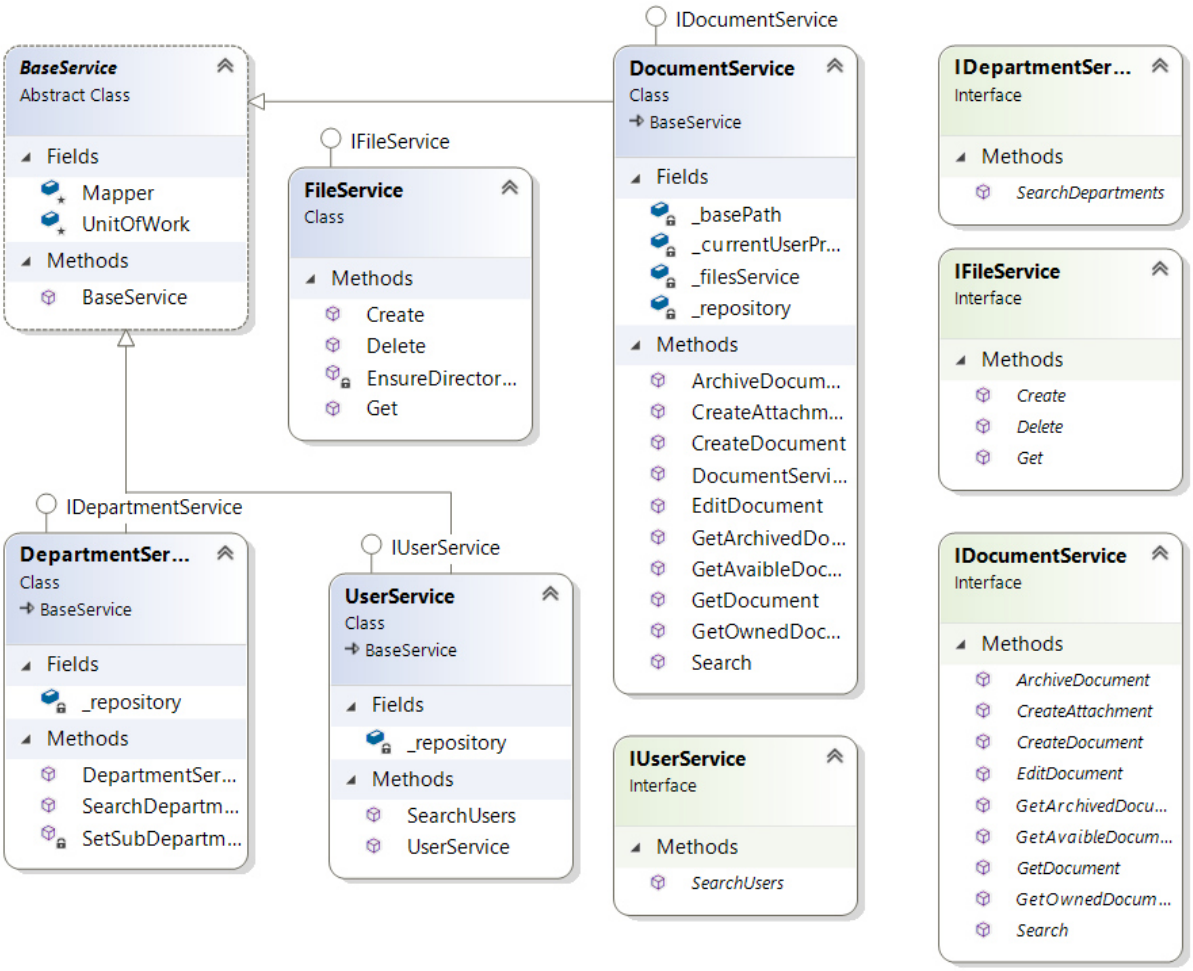


Рис. 3.4 Схема шару бізнес-логіки

Реалізоване WEB API включає в себе такі кінцеві точки:

- *GET /api/document/{id}* – отримання детальної інформації про документ, де id – це Id документу. Повертає статус-код 200 з відповідним Dto документу, якщо такий існує, або 404, якщо не існує. Схема DocumentDto зображена на Рис. 3.6.
- *GET /api/document/owned* - отримання інформації про документи, одним із власників яких є чинний користувач. Повертає статус-код 200 із списком відповідних Dto документів. Якщо документи у власності відсутні – повертає порожній список.

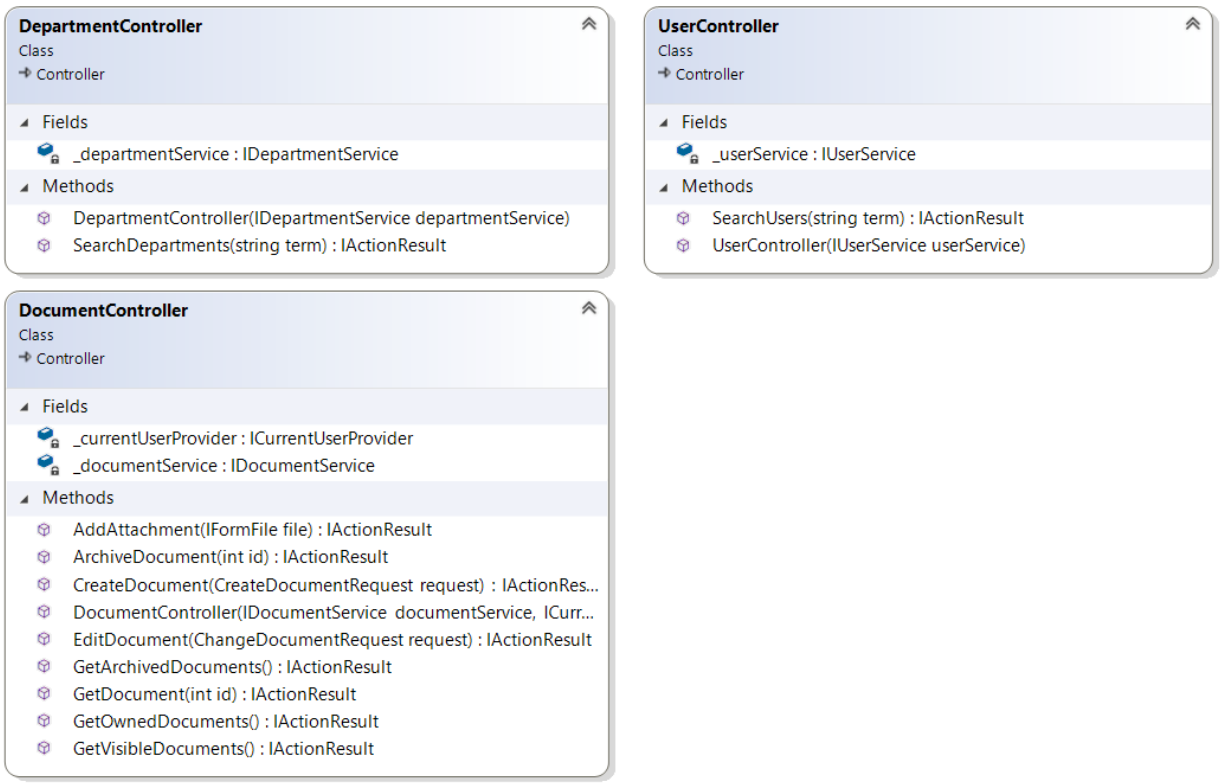


Рис 3.5 Схема шару WebApi

- *GET /api/document/visible* - отримання інформації про документи, доступом для читання яких володіє чинний користувач. Повертає статус-код 200 із списком відповідних Dto документів. Якщо документи у доступі для читання відсутні – повертає порожній список.

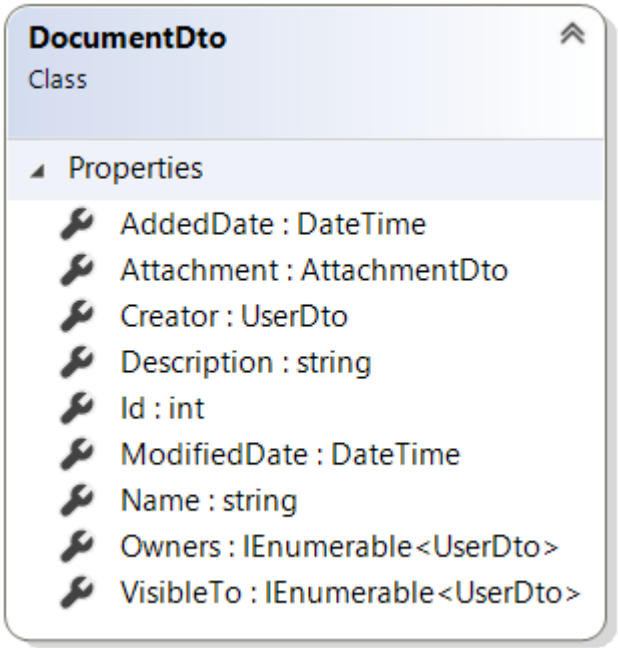


Рис. 3.6 DocumentDto

- *GET /api/document/archived* - отримання інформації про заархівовані документи, доступом до яких володіє чинний користувач. Повертає статус-код 200 із списком відповідних Dto документів. Якщо заархівовані документи у доступі відсутні – повертає порожній список.
- *POST /api/document* – створення нового документу. В частині Body http-запиту приймає Dto з інформацією про документ, що створюється. Його схему можна побачити на Рис. 3.7. При успішному створенні нового документу повертає статус-код 200.

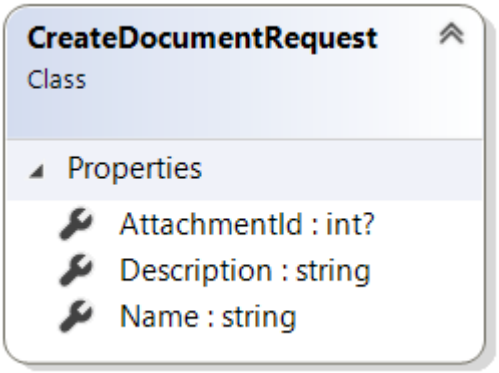


Рис. 3.7 CreateDocumentRequestDto

- *PUT /api/document* – зміна існуючого документу. В частині Body http-запиту приймає Dto з інформацією про документ, що редагується. Його схему можна побачити на Рис. 3.8. При успішній зміні документу повертає статус-код 200.

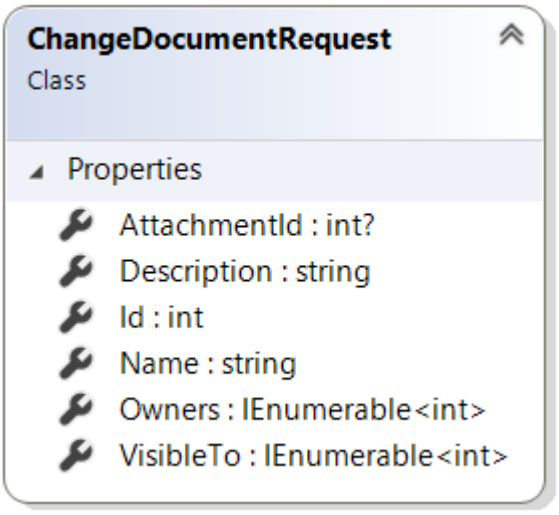


Рис. 3.8 ChangeDocumentRequestDto

- *POST /api/document/attachment* – додавання файлу до документу документу. В частині Body http-запиту приймає файл, що додається. При успішному додаванні файлу повертає статус-код 200 та Dto новоствореного аттачменту. Схема AttachmentDto зображена на Рис. 3.9.
- *GET /api/document/{id}* – архівування документу, де id – це Id документу. Повертає статус-код 200, якщо такий існує та його успішно заархівовано, або 404, якщо не існує.

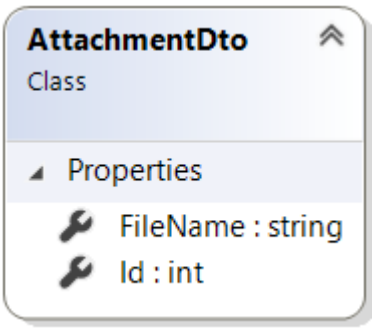


Рис. 3.9 AttachmentDto

- *GET /api/document/search/* – повнотекстовий пошук серед документів. В частині Body http-запиту приймає Dto з інформацією про пошуковий запит, який зображено на Рис. 3.12. Повертає статус-код 200 та список Dto знайдених результатів, які зображено на Рис. 3.13.

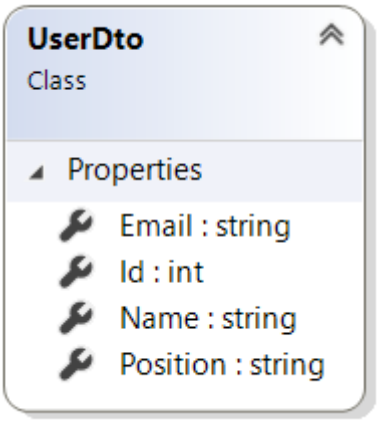


Рис. 3.10 UserDto

- *GET /api/user/search/{term}* – пошук по імені серед усіх користувачів, де term – це пошуковий запит. Повертає статус-код 200 та список Dto

знайдених користувачів. Якщо знайденої інформації немає – повертає порожній список. Схема UserDto зображена на Рис. 3.10.

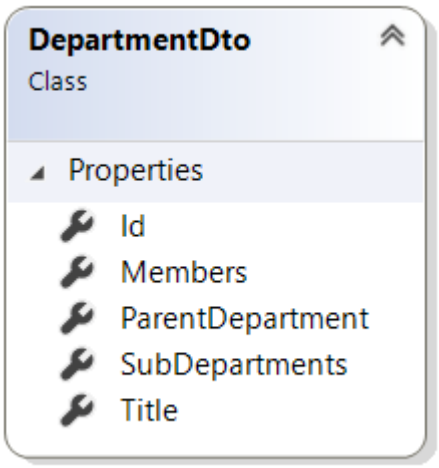


Рис. 3.11 DepartmentDto

- *GET /api/department/search/{term}* – пошук по імені серед усіх департаментів, де term – це пошуковий запит. Повертає статус-код 200 та список Dto знайдених департаментів. Якщо знайденої інформації немає – повертає порожній список.

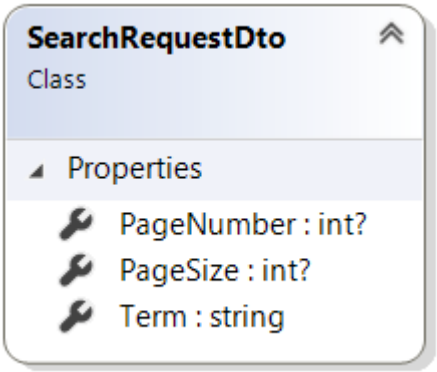


Рис. 3.12 SearchRequestDto

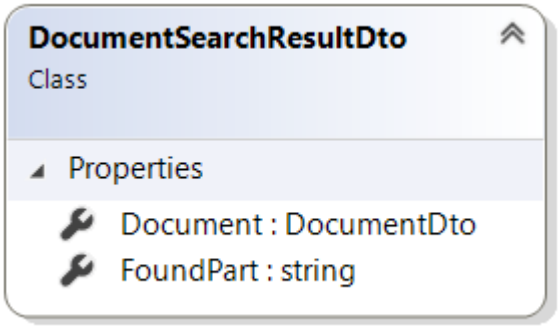


Рис. 3.13 DocumentSearchResultDto

3.3 Реалізація клієнтської частини додатку

Клієнтська частина являє собою SPA (Single Page Application) на основі фронт-енд фреймворку Angular 8. Структуру файлів проекту зображено на Рис. 3.14.

В сервісах реалізовані робота з WebAPI та спільна, специфічна для сутностей логіка. Наприклад в DocumentService окрім Http-запитів розміщений маппінг моделей пов’язаних з моделлю документу.

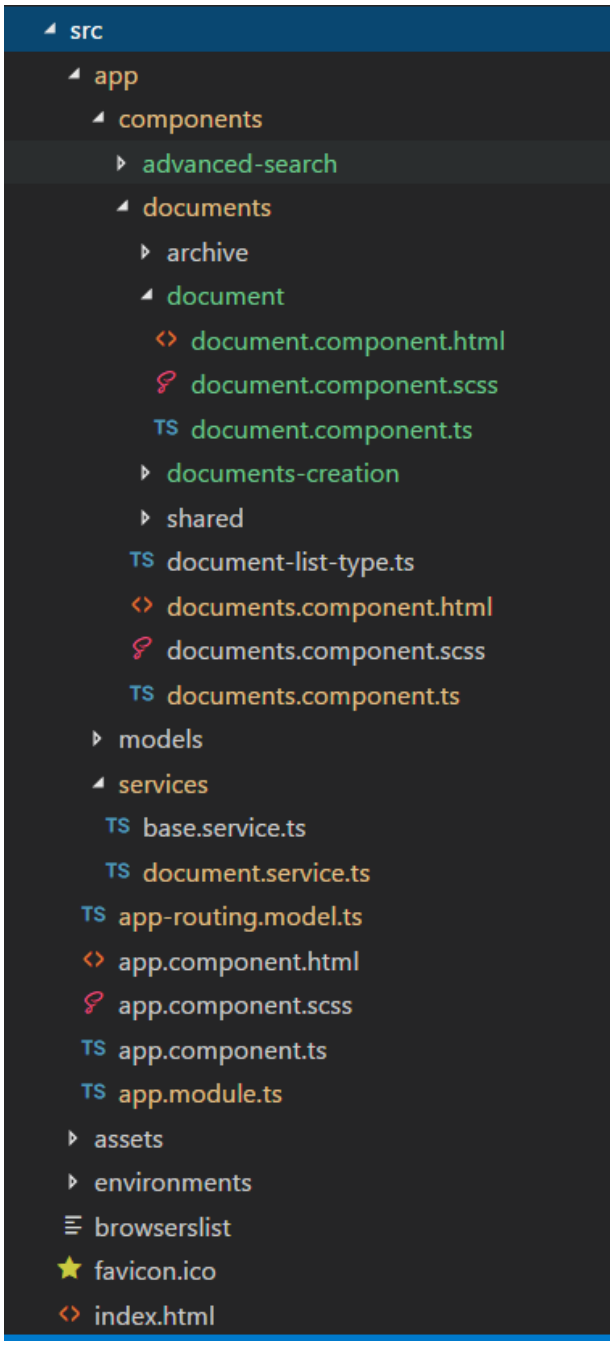


Рис. 3.14 Файлова структура клієнтської частини

Специфічну для сторінок логіку реалізовано в відповідних контроллерах.

В клієнтській частині реалізовано такі шляхи (routes):

- *‘documents’* – компонент: DocumentsComponent. Відкриває спільний компонент списку документів у режимі перегляду документів у власності.
- *‘shared’* – компонент: SharedComponent. Відкриває спільний компонент списку документів у режимі перегляду документів у власності.
- *‘archive’* - компонент: SharedComponent. Відкриває спільний компонент списку документів у режимі перегляду документів у власності.
- *‘document/:id’* – компонент: DocumentComponent. Відкриває сторінку документа.
- *‘advanced-search’* – компонент: AdvancedSearchComponent. Відкриває компонент повнотекстового пошуку
- *‘’* – перенаправлення на ‘documents’

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі представлено список конкретних технологій, що були використані для реалізації розроблюваного проекту. Також було обґрунтовано доцільність вибору.

Було представлено файлову структуру проекту, схему бази даних та описано деталі реалізації усіх шарів серверної та клієнтської частин додатку, надано відповідні схеми та діаграми.

Реалізований додаток повністю відповідає архітектурі, що була обрана та детально описана у другому розділі роботи.

РОЗДІЛ 4
ОГЛЯД РОЗРОБЛЕНГО ДОДАТКА

В даному розділі представлено демонстрацію розробленого додатку та зроблено загальний огляд його функціональних можливостей.

Розроблена система електронного документообігу являє собою веб-додаток. Скриншот головного вікна зображено на Рис. 4.1.

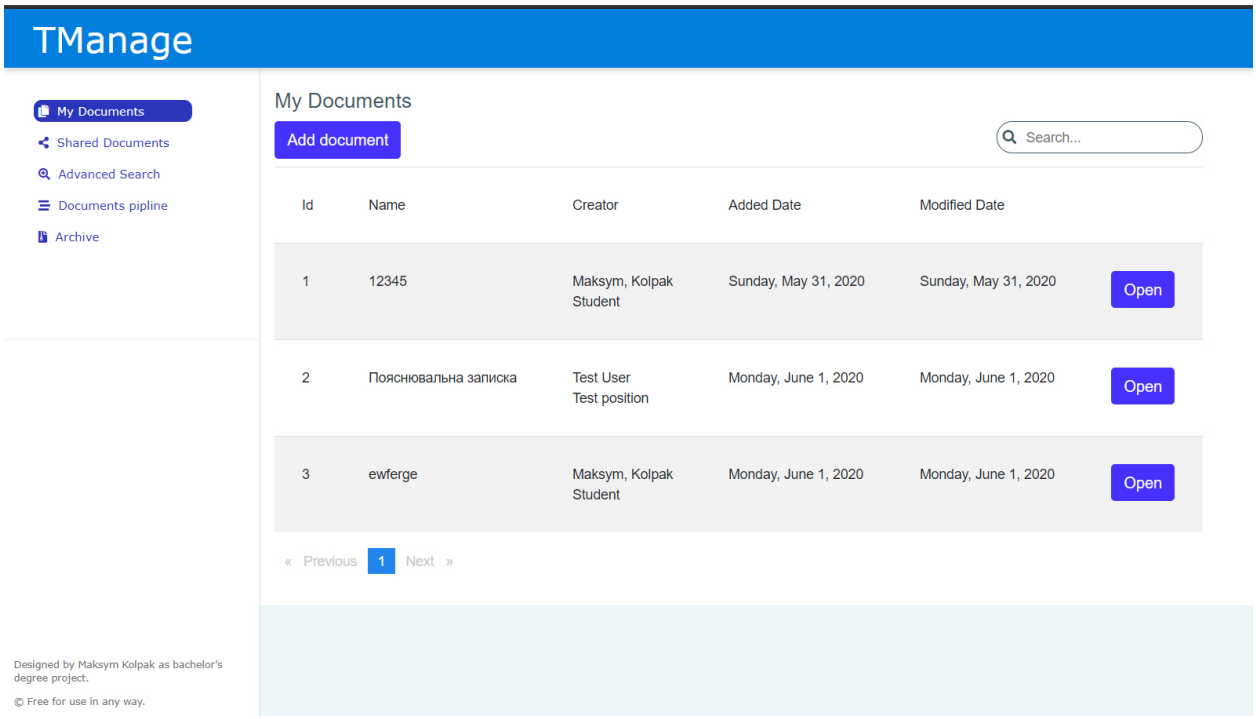


Рис. 4.1 Скриншот головного вікна додатку

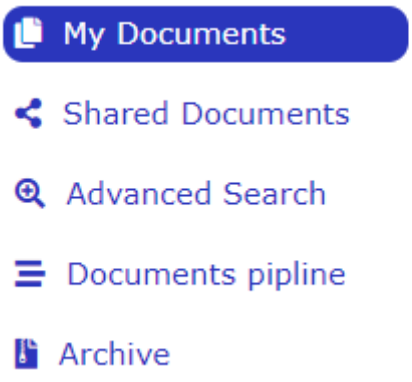


Рис. 4.2 Головне меню

Пункт ‘My Documents’ відкриває сторінку зі списком усіх документів, одним із власників яких є чинний користувач. Власник документу має право його редагувати, давати доступ для читання іншим користувачам, надавати права

власності над цим документом, редагувати його тощо. Скриншот цієї сторінки зображено на Рис. 4.3.

В кожному елементі списку вказано коротку інформацію про відповідний документ, а саме:

- Назва документу;
- Інформація про автора (його повне ім’я та посада);
- Дата створення документу;
- Дата останньої модифікації.

На даній сторінці працює простий пошук та пагінація.

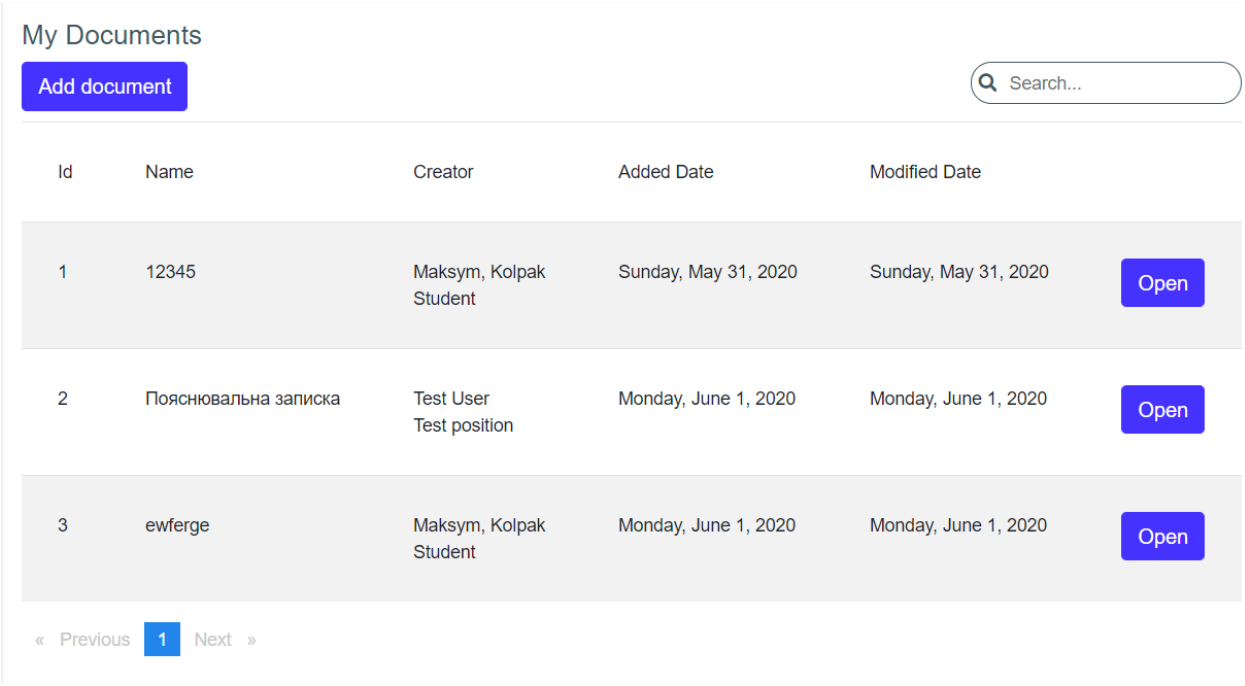


Рис. 4.3 Список власних документів

Кнопка ‘Add document’ відкриває модальне вікно з формою створення нового документу. Її скриншот представлено на Рис. 4.4. Тут міститься поле назви нового документа, поле опису документа та форма для завантаження файлу. Для створення документу заповнення усіх полів та завантаження файлу є обов’язковими, при недотриманні цієї вимоги з’явиться відповідне спливаюче повідомлення, яке зображено на Рис. 4.5. Автор документу після створення автоматично стає його першим власником.

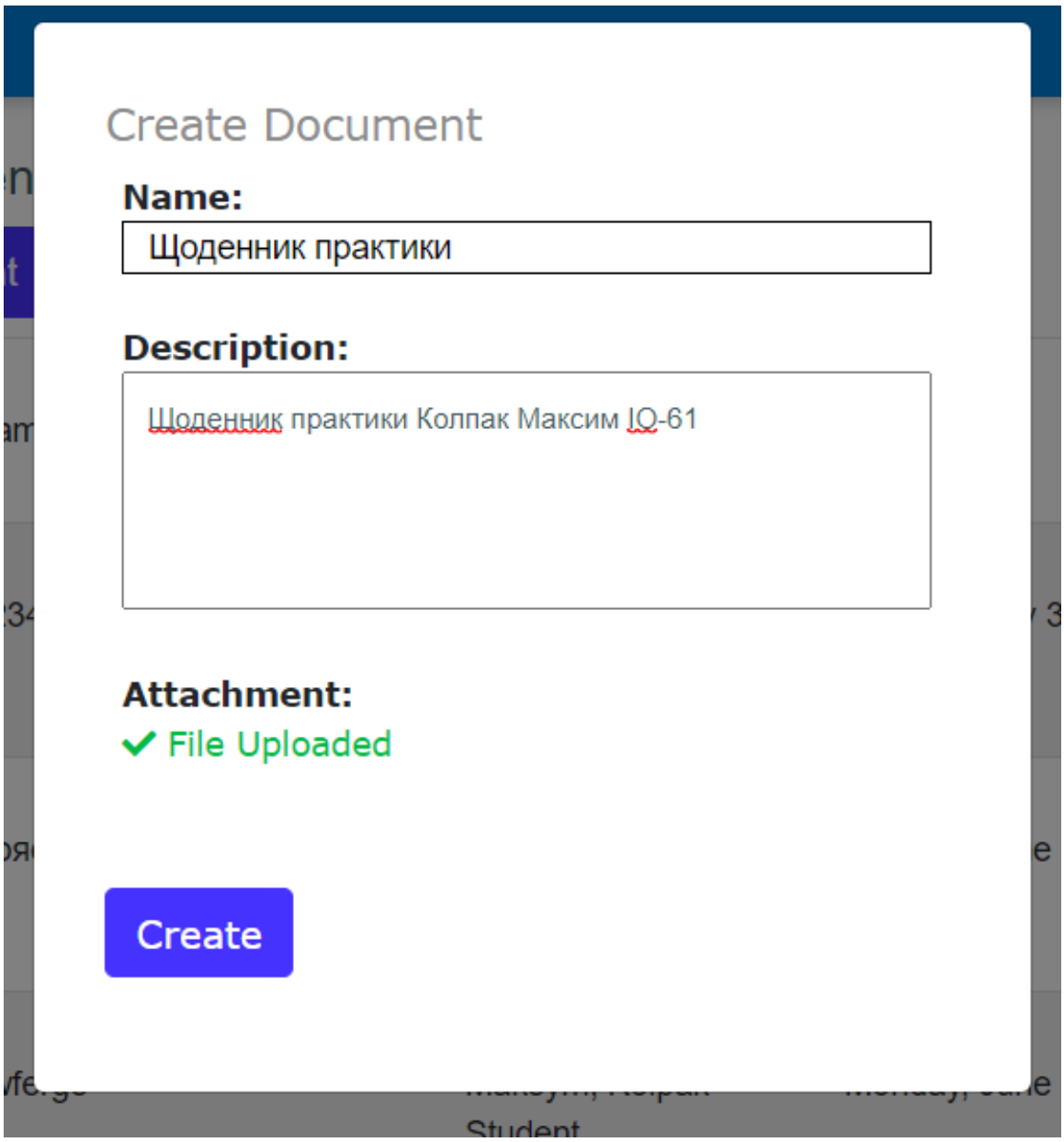


Рис. 4.4 Форма додавання документу

Рис. 4.5 Спливаюче повідомлення помилки в валідації

Пункт головного меню ‘Shared’ відкриває сторінку зі списком усіх документів, доступних теперішньому користувачу лише для читання. Відмінність від

сторінки власних документів, окрім набору показаних елементів списку, полягає у відсутності форми створення нового документу, все інше, тобто формат запису, пошук та пагінація співпадають. Скриншот даної сторінки зображено на Рис. 4.6.

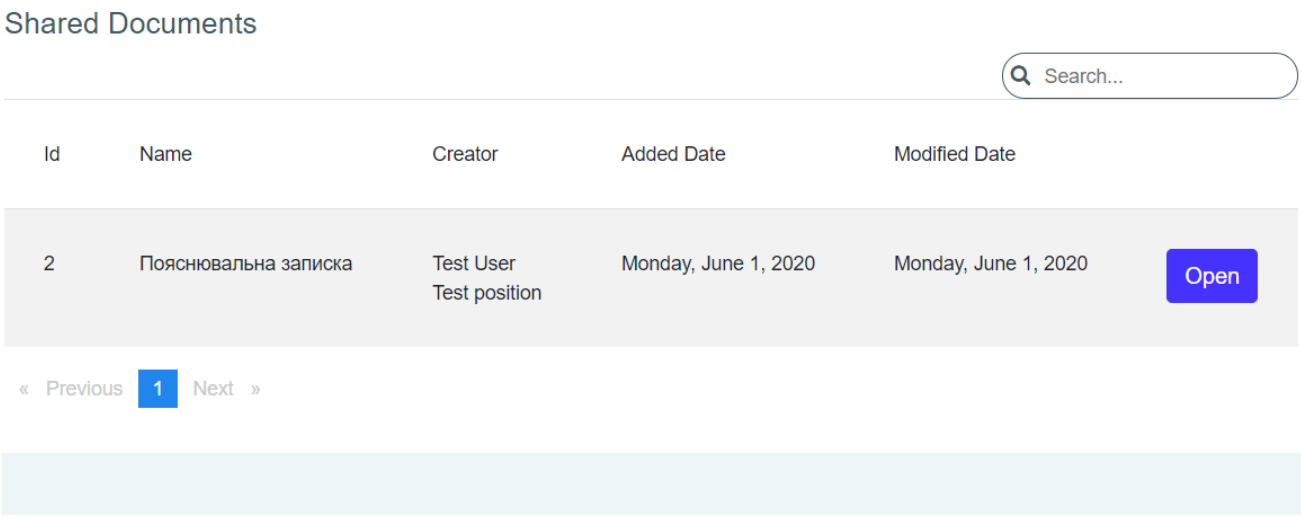


Рис. 4.6 Список доступних документів

Пункт головного меню ‘Archived’ відкриває сторінку зі списком усіх заархівованих документів, будь-яким чином пов’язаних з теперішнім користувачем. Документ в архіві не має файлу – тіла документу, та зберігає лише метаінформацію. Також, редагування архівних документів заборонене. Інших відмінностей від сторінки документів, доступних лише для читання немає.

При натисканні на кнопку “Open” в одному із елементів списку на перерахованих вище сторінках головного меню, відкриється сторінка з детальною інформацією про документ. Її скриншот зображено на Рис. 4.7.

Повна інформація включає в себе:

- Назву документу;
- Детальний опис;
- Інформацію про автора (повне ім’я та посада);
- Посилання на відповідний файл документу;
- Дата створення документу;
- Дата останньої зміни;
- Повний список людей, що мають доступ до документу (повні імена та посади).

Якщо чинний користувач є власником документу, то в нього з’являються додаткові елементи:

- Кнопка модифікації документа, що дозволяє змінювати деякі поля;
- Кнопка архівації документа;
- Список власників документа;
- Кнопки модифікації списків власників та людей з доступом для читання документу.

Скриншот сторінки власника документу зображено на Рис. 4.8

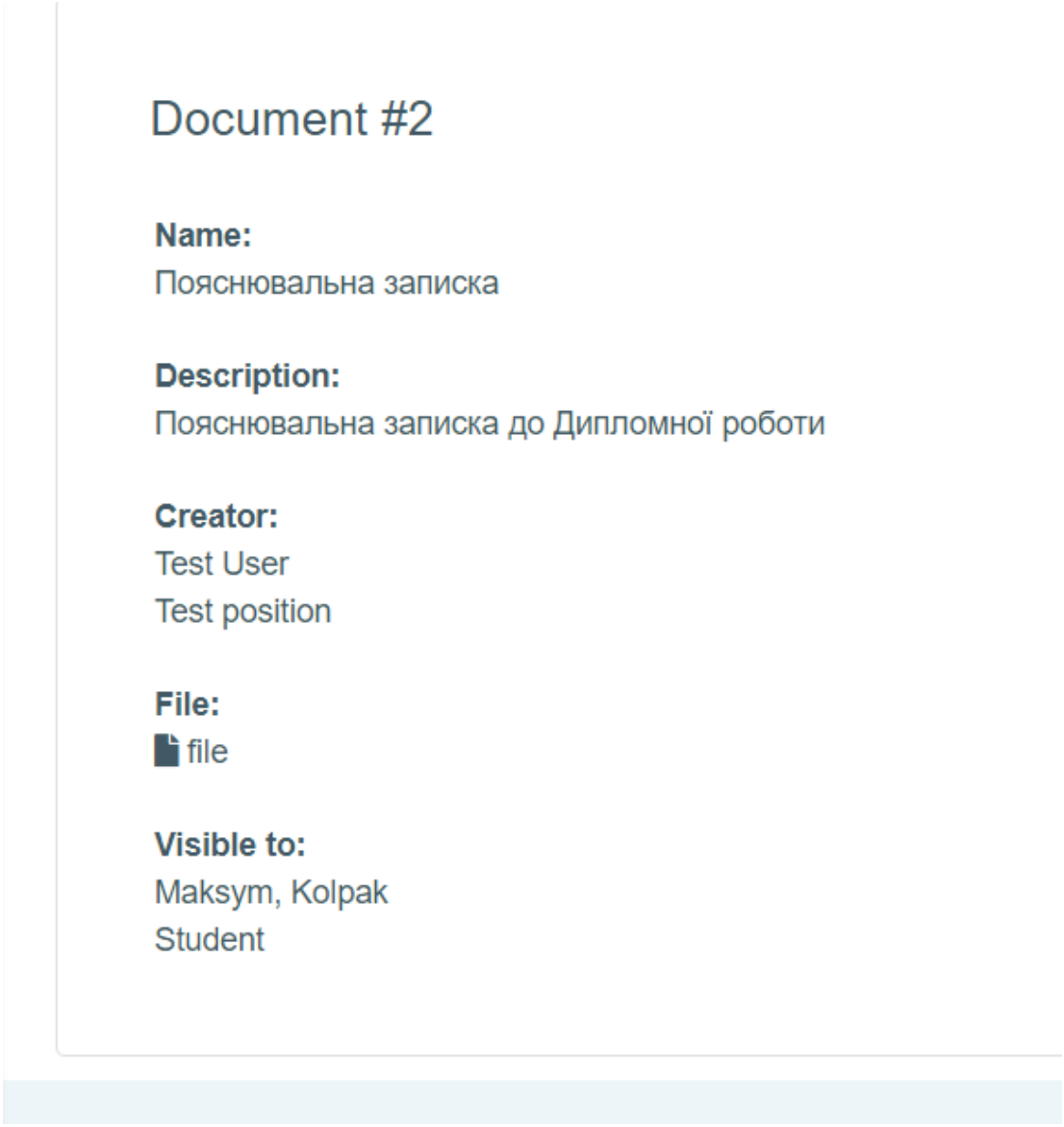


Рис. 4.7 Сторінка інформації про документ для читання

Додавання користувачів у списки документу відбувається в окремому модальному вікні. Власник може надавати та забирати доступ для читання у будь-якого користувача, який не є іншим власником. На відміну від цього, редагування списку власників доступне лише на додавання, тобто право власності на документ постійне та незмінне. Скриншоти обох видів модальних вікон зображено відповідно на Рис. 4.9 та 4.10.

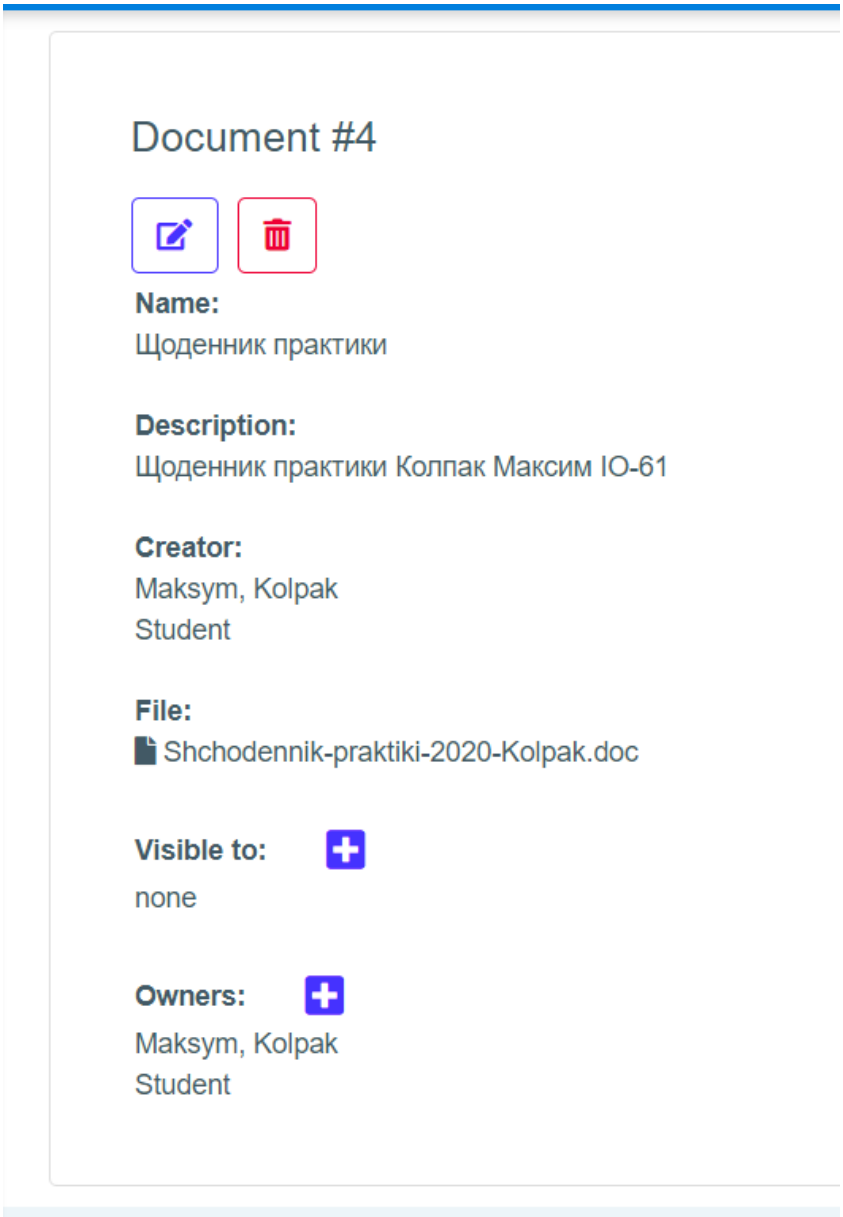


Рис. 4.8 Сторінка інформації про документ для власника

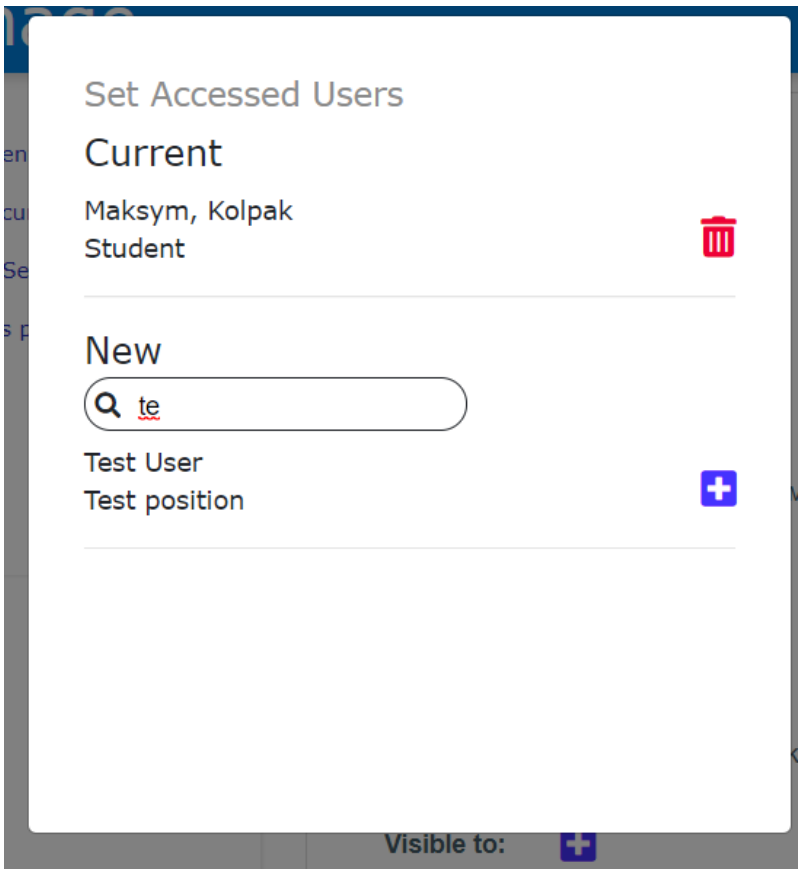


Рис. 4.9 Вікно редагування списку користувачів з доступом для читання

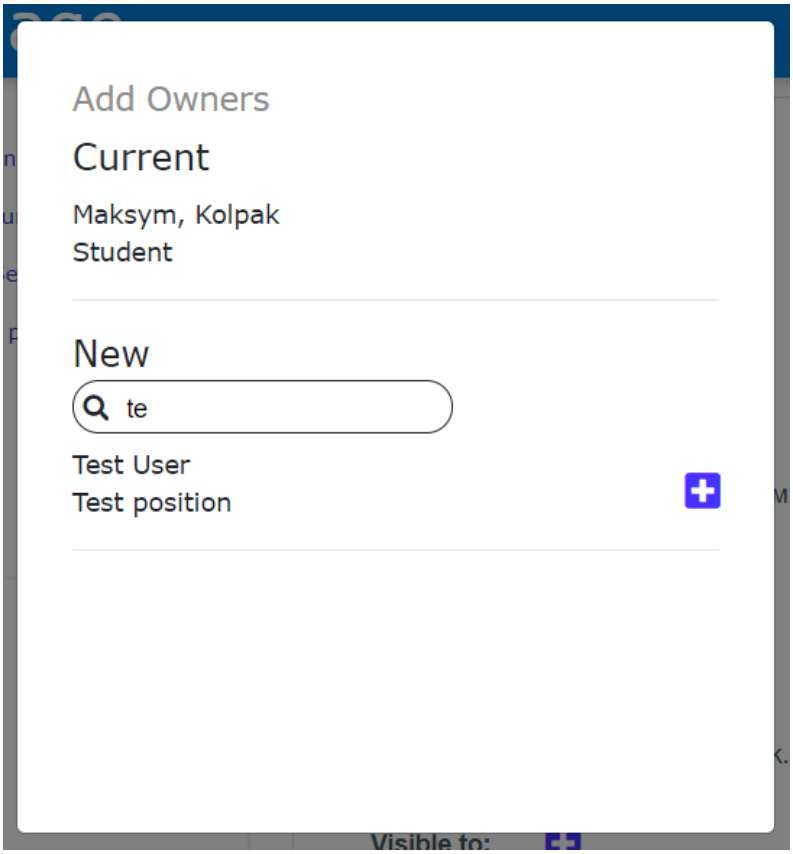


Рис. 4.10 Вікно додавання власників документу

Пункт головного меню ‘Advanced search’ відкриває сторінку повнотекстового пошуку по документам. Її скриншот зображено на Рис. 4.11. Наразі, повнотекстовий пошук працює лише з форматом файлів docx, в подальшому можливе додавання інших форматів. Індексація файлів відбувається 1 раз на добу, за розкладом. Можливий ручний запуск індексації, проте вона займе досить тривалий час.

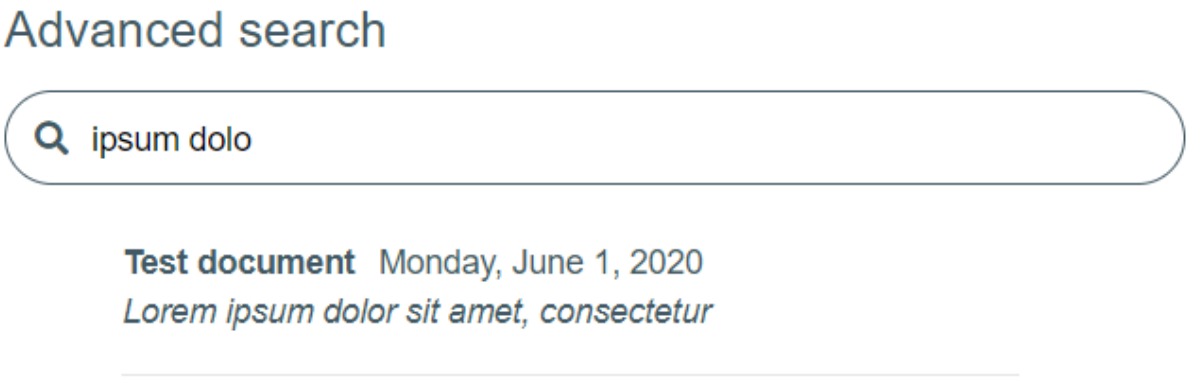


Рис. 4.11 Сторінка повнотекстового пошуку

ВИСНОВКИ ДО РОЗДІЛУ 4

Встановлено, що розроблений веб-додаток працює правильно та відповідає необхідним для нього вимогам.

Даний додаток дозволяє задовольнити базові потреби малих підприємств, надаючи такі функціональні можливості:

- Додавання та збереження документів;
- Редагування документів;
- Розділення прав доступу та володіння;
- Можливість надання прав на документ;
- Архівування;
- Повнотекстовий пошук.

При цьому система є безкоштовною та з відкритим програмним кодом, що було головною вимогою перед додатком, що розроблявся.

ВИСНОВКИ

В рамках даної роботи було проведено аналіз деяких представників ринку корпоративних систем електронного документообігу. Метою цього аналізу було знаходження проблем сучасних систем електронного документообігу, що можуть бути причиною відмови деяких підприємств від використання даних програмних продуктів. В результаті, такими проблемами виявились висока ціна та закритість програмного коду представлених систем.

Наступним завданням стала розробка системи, яка б змогла уникнути цих проблем, надавши базову функціональність повноцінної системи електронного документообігу.

Було проаналізовано сучасні архітектури програмного забезпечення на предмет придатності та доцільності використання в системах такого виду. В результаті було обрано багатошарову архітектуру.

Наступним етапом роботи стала власне розробка системи. Було імплементовано веб-додаток на базі ASP.NET Core 3.1 та Angular 8.

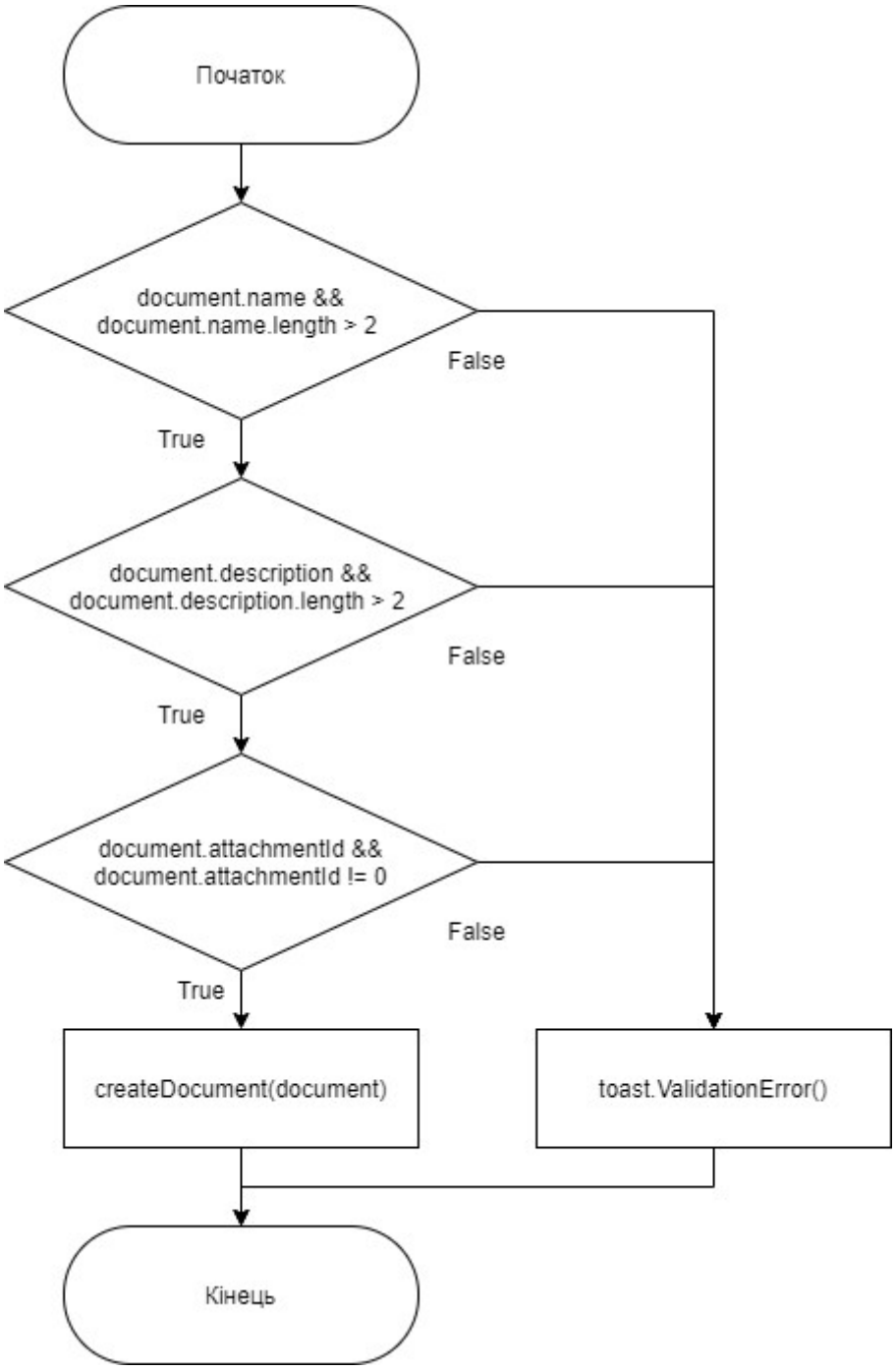
Розроблений додаток повністю виконав покладені на нього задачі.

Під час виконання проекту було використано та поглиблено знання та навички у сфері проектування та розробки програмного забезпечення.

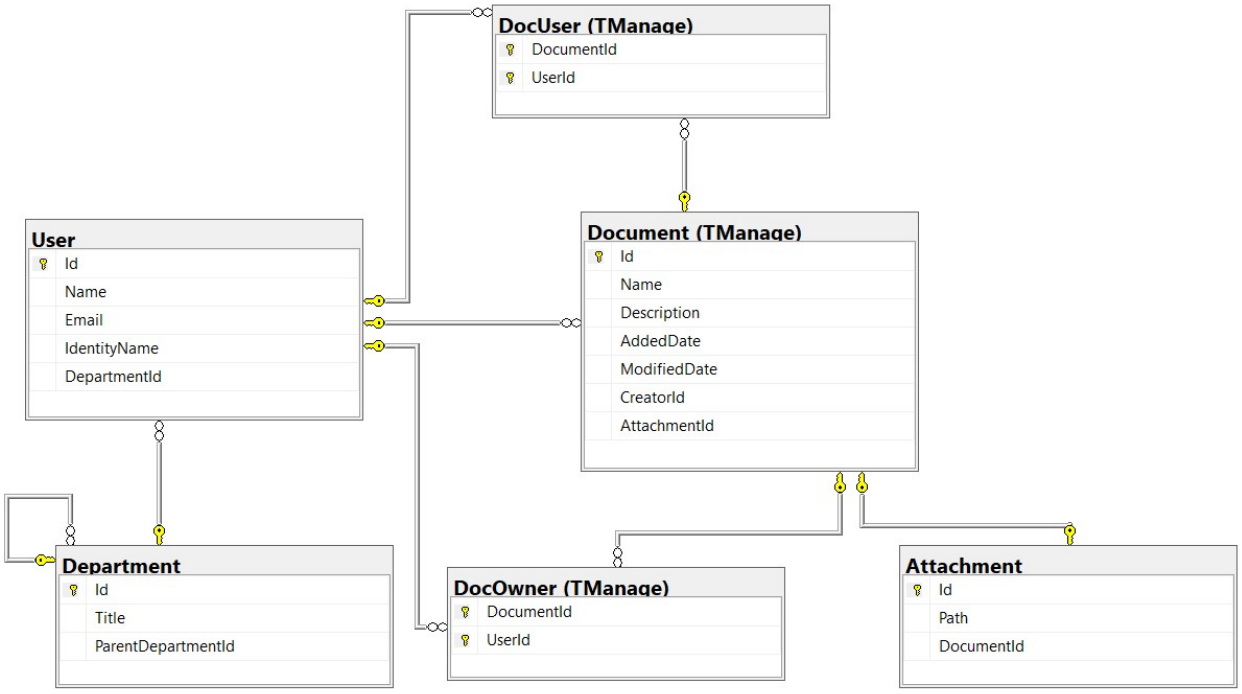
ПЕРЕЛІК ПОСИЛАНЬ

1. Valerio De Sanctis ASP.NET Core 3 and Angular 9. Third Edition. Full stack web development with .NET Core 3.1 and Angular 9 / Valerio De Sanctis // Birmingham, - Packt Publishing Ltd. - 2020. P. 724.
2. Дино Эспозито Microsoft .NET: архитектура корпоративных приложений / Дино Эспозито, Андреа Сальтарелло //
3. Чамберс Джеймс ASP.NET Core. Разработка приложений / Чамберс Джеймс, Пэкетт Дэвид, Тиммс Саймон // – СПБ. – Питер, 2018. – 464с.
4. FossDoc [Електронний ресурс]. – ФОСС-Он-Лайн, 1999-2018. – Режим доступу: *https://fossdoc.com/*.
5. Holger Schwichtenberg Modern Data Access with Entity Framework Core. Database Programming Techniques for .NET, .NET Core, UWP, and Xamarin with C# / Holger Schwichtenberg // - Essen, Germany. - Appress, 2018. - P. 644.
6. Джефри Рихтер CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. // Рихтер Дж. // — СПб.: Питер, 2013. — 896 с.
7. Docsvision [Електронний ресурс]. - ДоксВижн, 2020. – Режим доступу: *https://e-docs.ua/*.
8. E-Docs [Електронний ресурс]. – e-Docs, 2020. – Режим доступу: *https://www.docsvision.com/*.
9. Prozorro публічні закупівлі [Електронний ресурс]. – ДП "ПРОЗОРРО", 2019. – Режим доступу: *https://prozorro.gov.ua/tender/UA-2019-07-29-000305-c*.
10. Docassemble [Електронний ресурс]. - Jonathan Pyle – Режим доступу: *https://docassemble.org/docs.html*.
11. ФОСС-Он-Лайн [Електронний ресурс]. – Режим доступу: *https://community.foss.kharkov.ua/viewtopic.php?f=59&t=773&start=30/*.

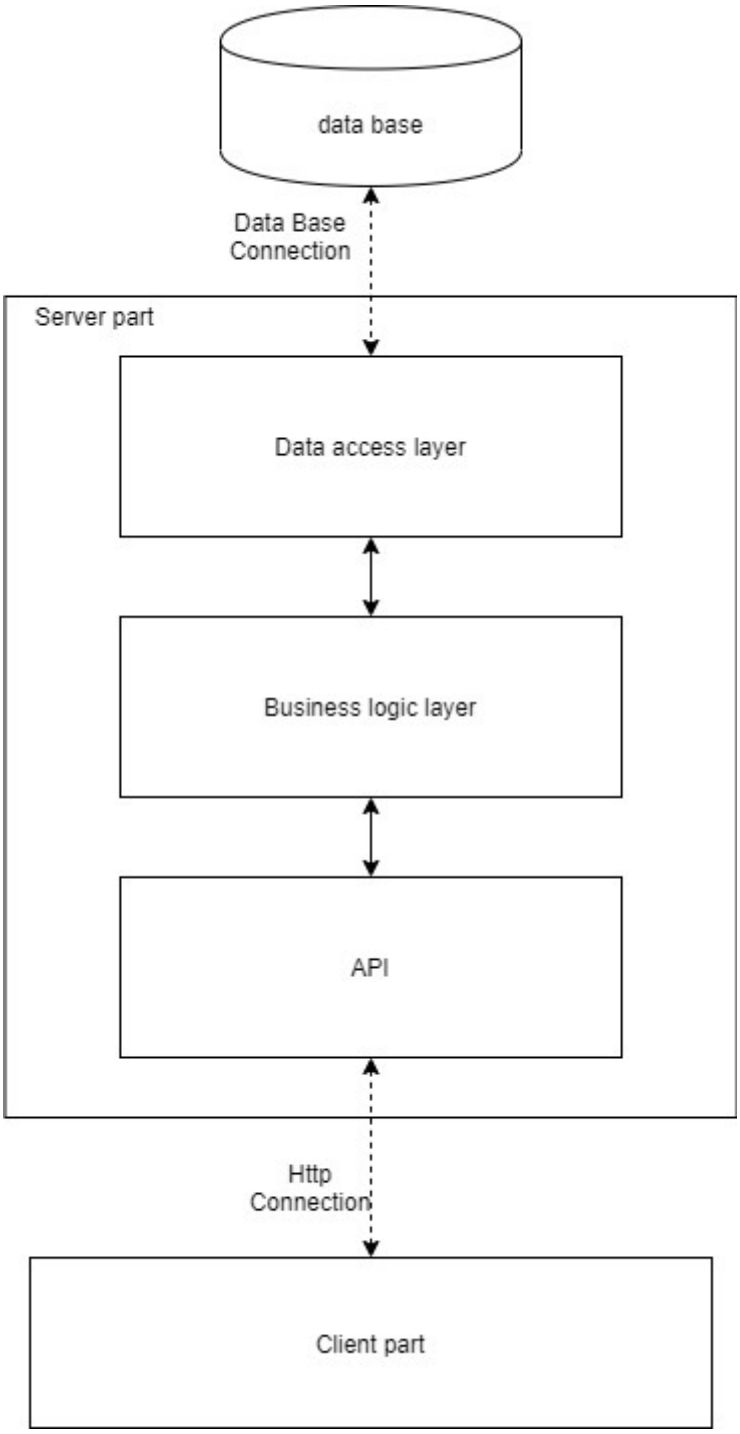
12. ASP.NET Core Middleware [Електронний ресурс]. – Microsoft, 2020 – Режим доступу: *https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-3.1*.
13. Angular Docs [Електронний ресурс]. – Google, 2010-2020. – Режим доступу: *https://angular.io/guide/architecture*.
14. Выбор СЭД — о чем молчат вендоры [Електронний ресурс]. – ТМ, 2013. – Режим доступу: *https://habr.com/ru/post/198536/*.
15. Hangfire, - Sergey Odinokov, 2013-2020. – Режим доступу: *https://docs.hangfire.io/en/latest/*.
16. Swagger [Електронний ресурс]. – SmartBear Software, 2020. – Режим доступу до ресурсу: *https://swagger.io/*.



					ІАЛЦ. 467200.003 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Колпак М.В.			Система електронного документообігу підприємства Алгоритм валідації форми створення документу	Літ. / Аркуш: 1 / Аркушів: 1
Перевірив		Сергієнко А.М.				
Реценз.						НТУУ КПІ, ФІОТ, ІО-61
Н. Контр.		Сімоненко В. П.				
Затвердив						



					ІАЛЦ. 467200.004 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Колпак М.В.			Система електронного документообігу підприємства Діаграма бази даних	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Сергієнко А.М.				
Реценз.						НТУУ КПІ, ФІОТ, ІО-61
Н. Контр.		Сімоненко В. П.				
Затвердив						



					ІАЛЦ. 467200.005 Д3	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Колпак М.В.			Система електронного документообігу підприємства Структурна схема системи	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Сергієнко А.М.				
Реценз.						НТУУ КПІ, ФІОТ, ІО-61
Н. Контр.		Сімоненко В. П.				
Затвердив						